Introduction of AI (CS103)- 13
BP and Deep Learning

Jimmy Liu 刘江

2023-12-15

Lecture 13

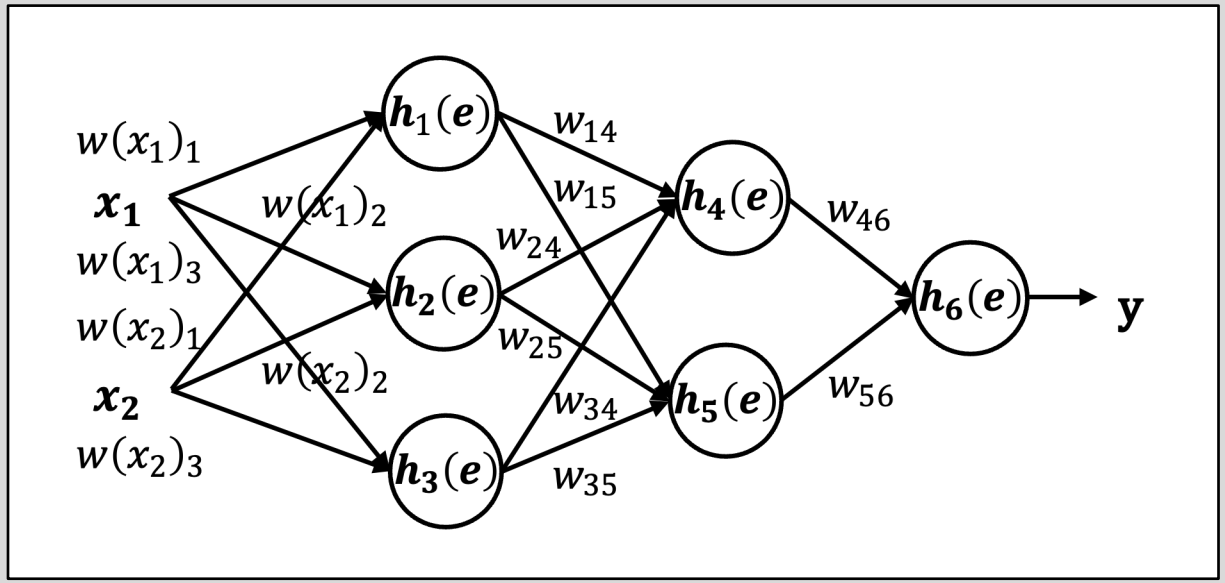
1. Reviews of Lecture 12
2. Deep Learning- Back Propagation
3. Deep Learning- Forward Neural Networks
4. Deep Learning- Transformer
5. Deep Learning- CNN

Gemini

Google CEO, Sundar Pichai, has announced the dawn of a new age in artificial intelligence (AI) — the Gemini era. Unveiled at the I/O developer conference and now available to the public, Gemini is Google's latest large language model (LLM), set to revolutionize Google's product suite.

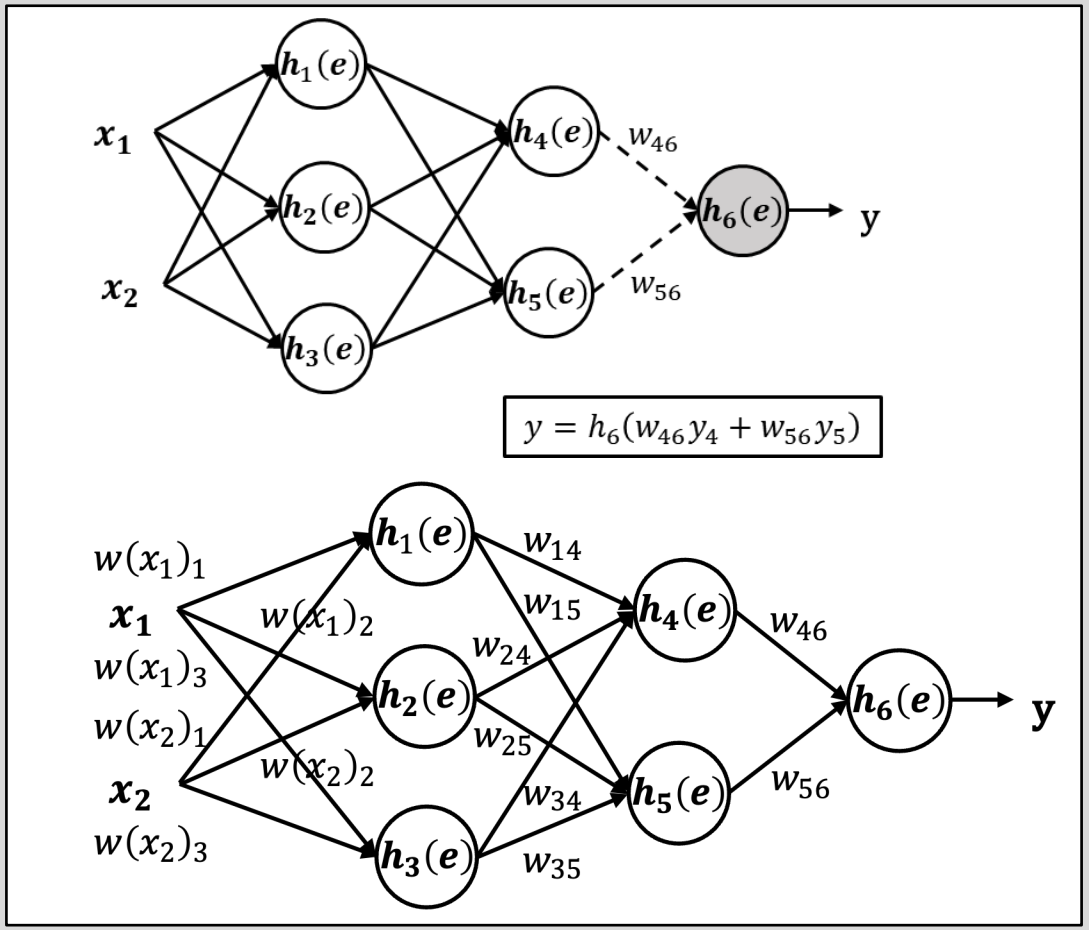
Gemini is not a single AI model. It's a versatile technology suite, including a streamlined version, Gemini Nano, designed for native, offline use on Android devices. A more robust variant, Gemini Pro, is set to power many Google AI services and is already the backbone of Bard. The most potent model, Gemini Ultra, designed for data centers and enterprise applications, is slated for release next year. The Gemini era begins with Bard, now powered by Gemini Pro, while Pixel 8 Pro users will enjoy new features courtesy of Gemini Nano. Developers and enterprise customers can access Gemini Pro via Google Generative AI Studio or Vertex AI in Google Cloud from December 13th.

Neural Network Training Example



To obtain satisfactory parameters for a neural network, we typically need to train it using labeled data. The given dataset consists of input samples (x_1 and x_2) and their corresponding labels (expected output $\hat{y}$). Neural network training is an **iterative** process. In each iteration, the weights of the nodes are modified using new data from the dataset. Here, we use this multi-layer neural network as an example to explain the backpropagation algorithm for incentive propagation and weight updating.

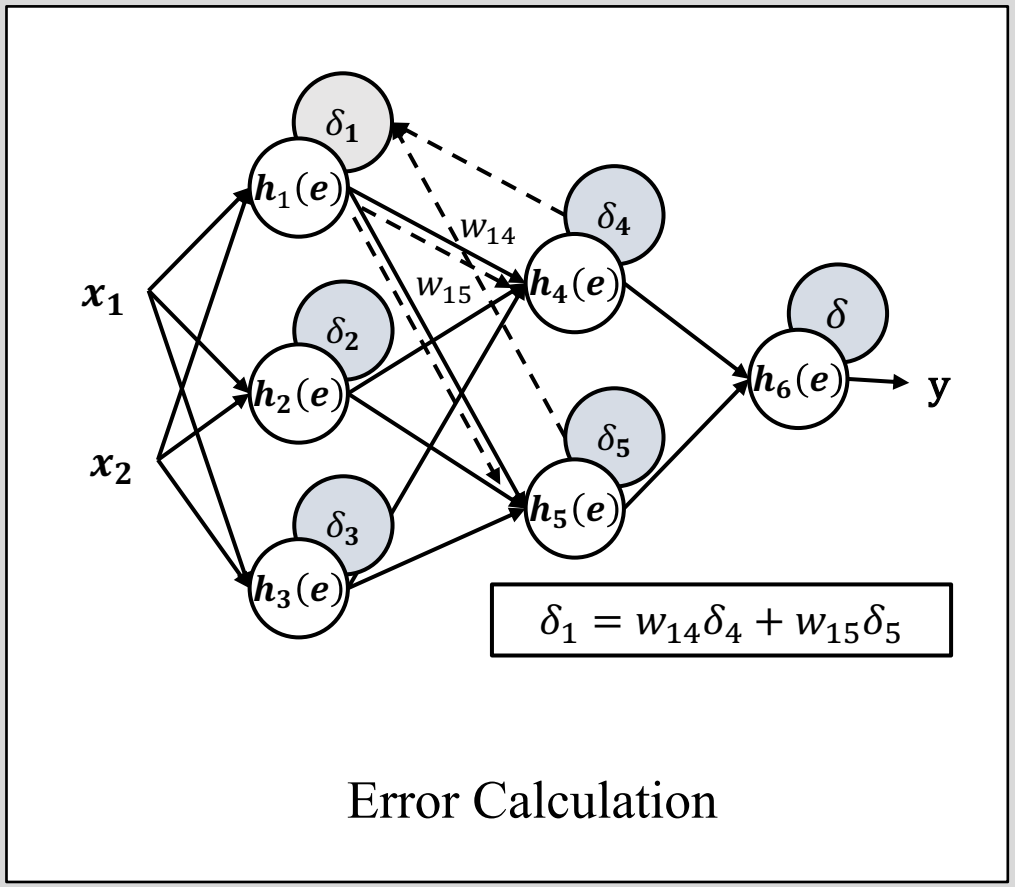
Forward Propagation



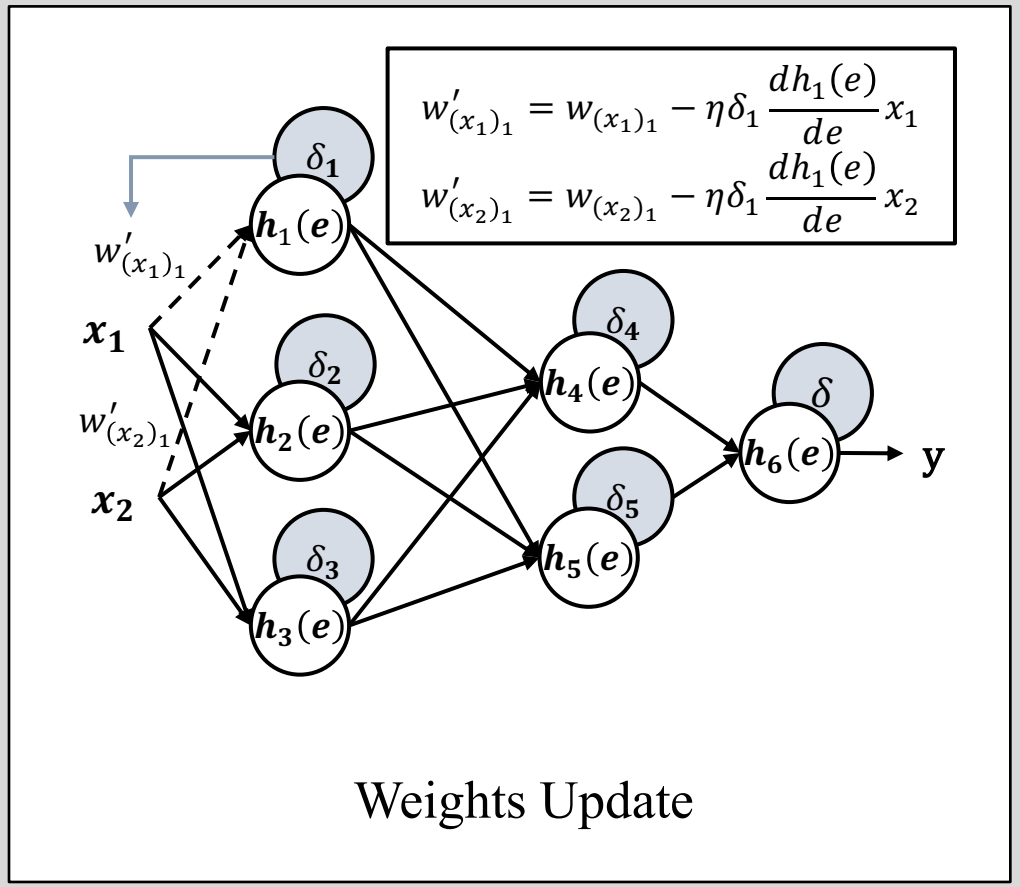
The symbol w_{mn} represents the connection weight between neuron m's output and neuron n's input in the next layer. As the signals pass through each neuron, the output signals are respectively:

$$y_1 = h_1\left(w_{(x_1)_1}x_1 + w_{(x_2)_1}x_2\right)$$
$$y_2 = h_2\left(w_{(x_1)_2}x_1 + w_{(x_2)_2}x_2\right)$$
$$y_3 = h_3\left(w_{(x_1)_3}x_1 + w_{(x_2)_3}x_2\right)$$
$$y_4 = h_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$
$$y_5 = h_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$
$$y = h_6(w_{46}y_4 + w_{56}y_5)$$

Backward Propagation (Bottom to Top)



Error Calculation



Weights Update

Weight Updating

$$w'_{(x_1)_1} = w_{(x_1)_1} - \eta\delta_1 \frac{dh_1(e)}{de} x_1, \quad w'_{(x_2)_1} = w_{(x_2)_1} - \eta\delta_1 \frac{dh_1(e)}{de}$$

$$w'_{(x_1)_2} = w_{(x_1)_2} - \eta\delta_2 \frac{dh_2(e)}{de} x_1, \quad w'_{(x_2)_2} = w_{(x_2)_2} - \eta\delta_2 \frac{dh_2(e)}{de} x_2$$

$$w'_{(x_1)_3} = w_{(x_1)_3} - \eta\delta_3 \frac{dh_3(e)}{de} x_1, \quad w'_{(x_2)_3} = w_{(x_2)_3} - \eta\delta_3 \frac{dh_3(e)}{de} x_2$$

$$w'_{14} = w_{14} - \eta\delta_4 \frac{dh_4(e)}{de} y_1, \quad w'_{24} = w_{24} - \eta\delta_4 \frac{dh_4(e)}{de} y_2, \quad w'_{34} = w_{34} - \eta\delta_4 \frac{dh_4(e)}{de} y_3$$

$$w'_{15} = w_{15} - \eta\delta_5 \frac{dh_5(e)}{de} y_1, \quad w'_{25} = w_{25} - \eta\delta_5 \frac{dh_5(e)}{de} y_2, \quad w'_{35} = w_{35} - \eta\delta_5 \frac{dh_5(e)}{de} y_3$$

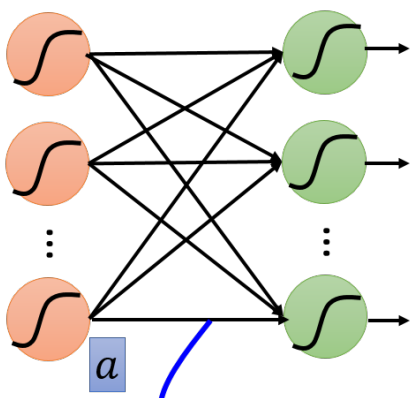
$$w'_{46} = w_{46} - \eta\delta \frac{dh_6(e)}{de} y_4, \quad w'_{56} = w_{56} - \eta\delta \frac{dh_6(e)}{de} y_5$$

Backpropagation Algorithm

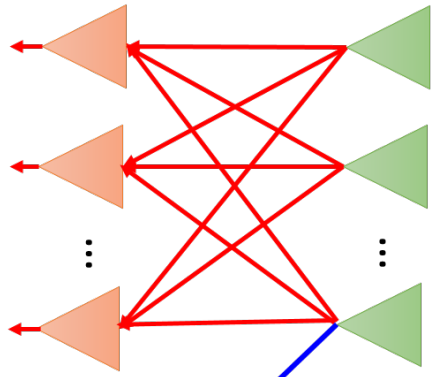
Algorithm 2 Back Propagation (BP) Algorithm

Input: dataset $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, learning rate η
Output: W
1: randomly initialize W
2: **repeat**
3: reorder the samples in dataset D
4: **for** $n = 1 \ldots$ N **do**
5: select sample $(x^{(n)}, y^{(n)})$ from the dataset D
6: calculate the output of each layer forward until the last layer
7: back propagation calculates the error of each layer and updates the parameters
8: **end**
9: **until** the neural network reaches a certain termination condition

Q1: Does Backpropagation Align with Human Learning ?



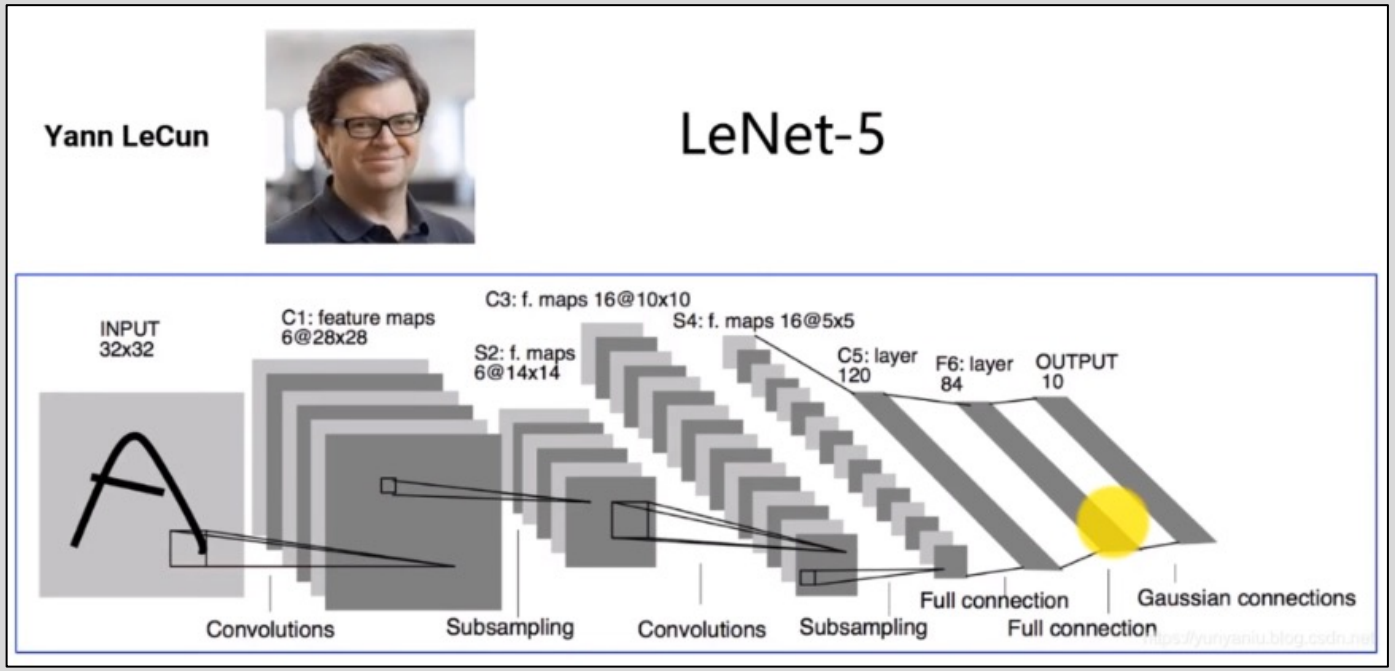
$$\frac{\partial z}{\partial w} = a$$



$$\text{X} \quad \frac{\partial l}{\partial z} \quad = \frac{\partial l}{\partial w}$$

for all w

Forward Neural Network: LeNet



In 1998, Yann LeCun, the "father of convolutional networks", implemented a seven-layer convolutional neural network LeNet-5 for the first time, which achieved superior performance in handwritten digital tasks.

Forward Neural Network: LeNet

AlexNet 网络结构

AlexNet network architecture

In 2012, AlexNet designed by Hinton and his student Alex Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a huge advantage, which triggered a new wave of deep learning research. It is the first time to apply techniques such as ReLU, Dropout and local normalization in the convolutional neural network, and at the same time use GPU parallel computing to accelerate the calculation of the network model, which has become the standard configuration for subsequent deep neural network design.

Forward Neural Network: AlexNet

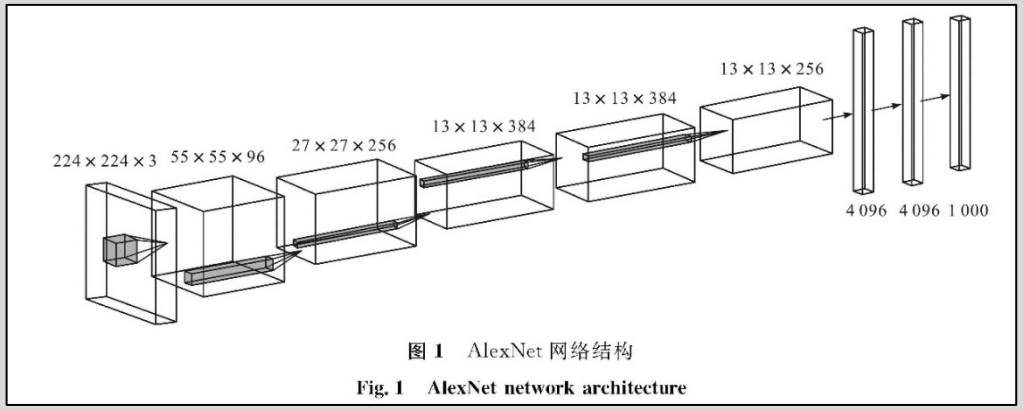
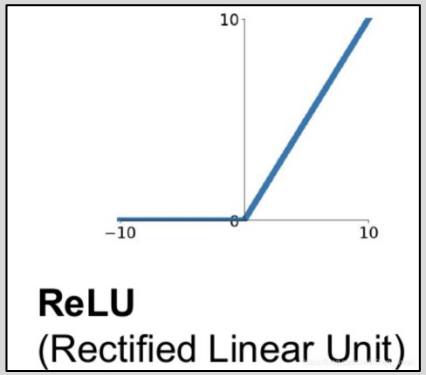
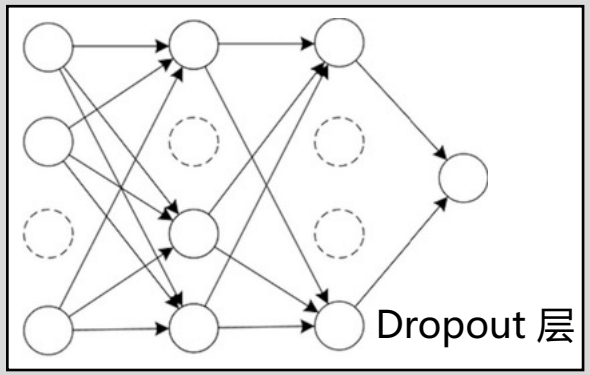


图 1　AlexNet 网络结构

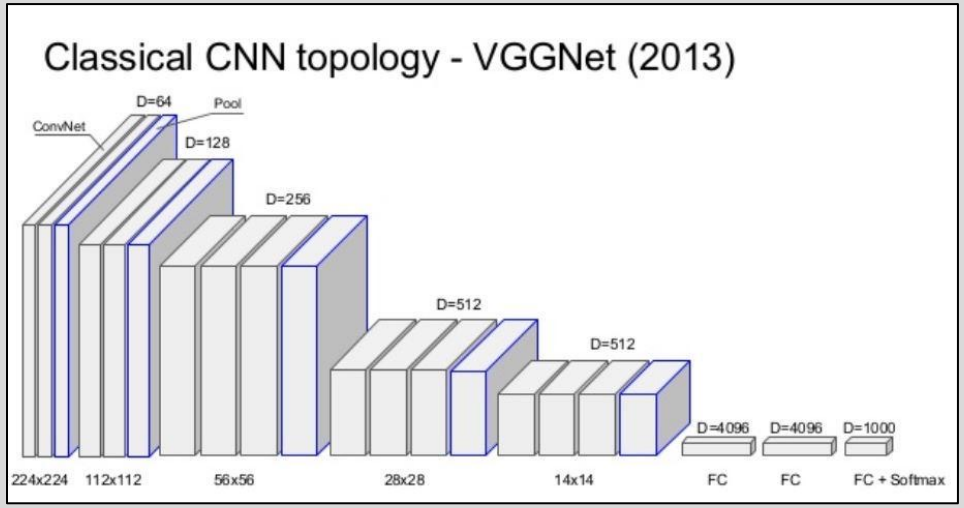
Fig. 1　AlexNet network architecture





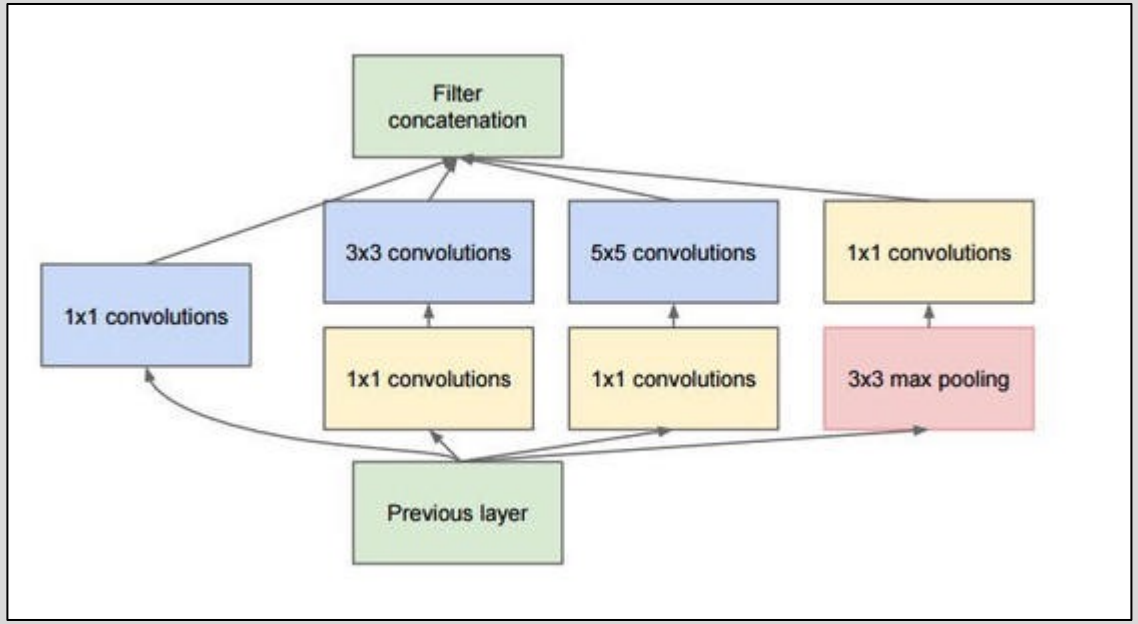
For the first time, techniques such as **ReLU**, **Dropout**, and **local normalization** were applied to convolutional neural networks, and GPU parallel computing was used to accelerate the calculation of network models. These have become standard features in subsequent deep neural network designs.

Forward Neural Network: VGGNet



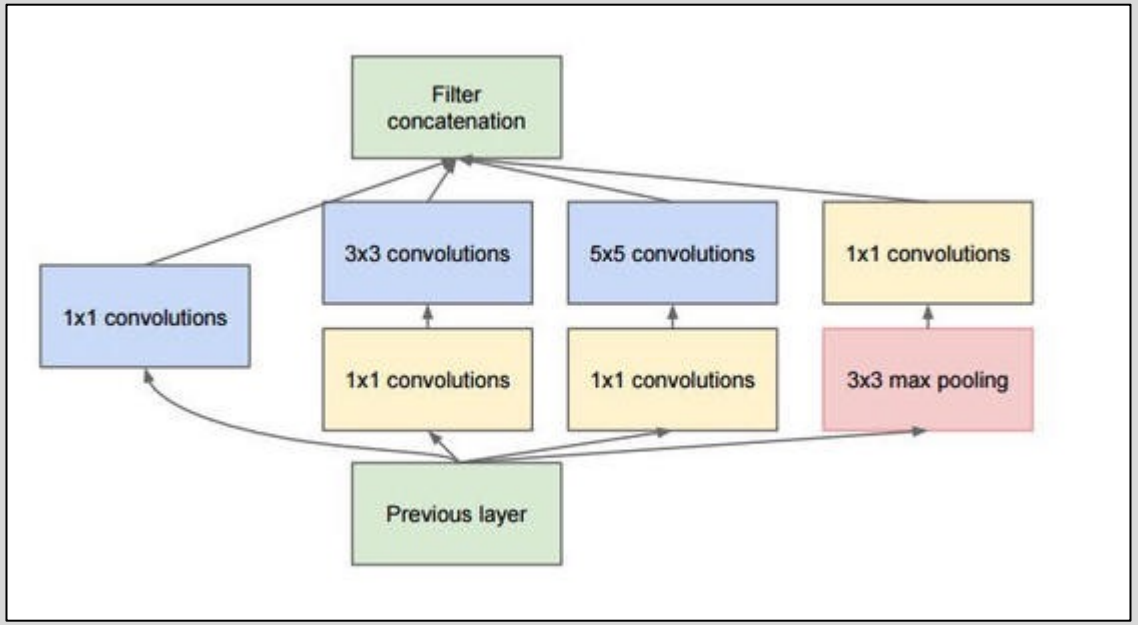
- VGGNet was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group (VGG) of Oxford University. VGGNet won first place in the positioning task and second place in the classification task in ILSVRC-2014.
- Its main innovation is the use of small 3*3 convolutional filters in each layer, with a relatively small receptive field. It allows building deeper networks while keeping the number of parameters manageable.

Forward Neural Network: GoogLeNet



GoogLeNet is a new deep learning structure proposed by Christian Szegedy at the ILSVRC Challenge in 2014. Previous structures such as AlexNet and VGG achieved better training results by increasing the depth (number of layers) of the network. However, the increase in the number of layers will bring many negative effects, such as overfit, gradient disappearance, gradient explosion, etc. The introduction of inception improves training results from another perspective: it can use computing resources more efficiently and extract more features with the same amount of calculation, thereby improving training results.

Forward Neural Network: GoogLeNet



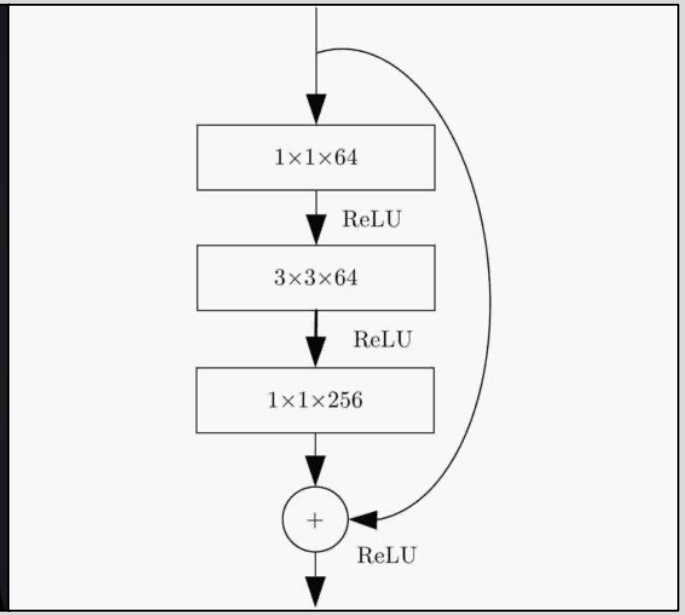
Inception module

The picture on the left shows the basic mechanism of the inception module, the entire GoogleNet structure is connected in series by multiple such inception modules. There are two main contributions of the inception structure:

- Use 1x1 convolution to increase and decrease dimensions;
- Perform convolution and re-aggregation on multiple dimensions simultaneously.

Forward Neural Network: ResNet

Kaiming He



ResNet

In 2016, the residual network (ResNet) proposed by He Kaiming and others won the championship of image classification and object recognition in the ImageNet large-scale visual recognition competition.

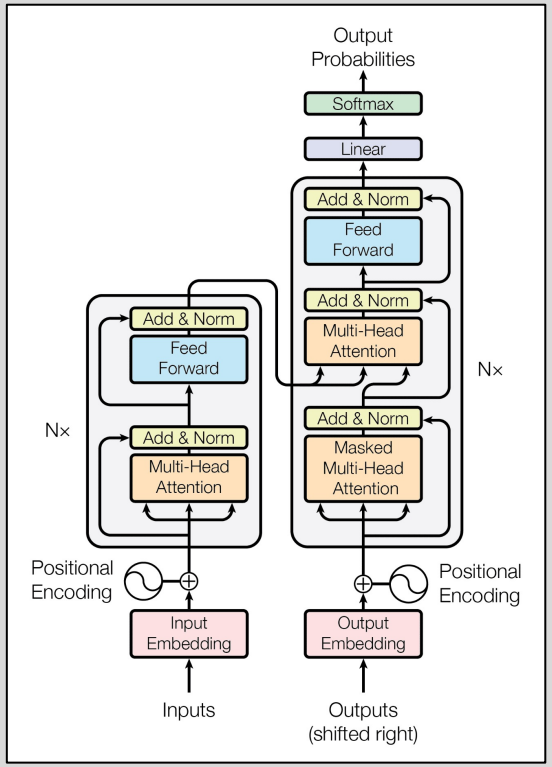
Forward Neural Network: ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$

Advantage：

- Ultra-deep network structure: the stronger the representation ability of the network.
- Proposed residual module: prevent network degradation in deeper networks. Vanishing or exploding gradients.

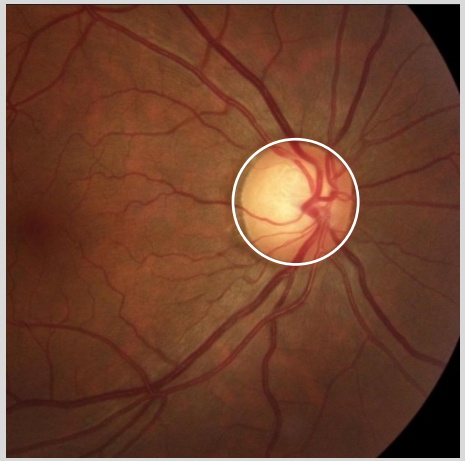
Transformer



In 2017, Vaswani, a researcher on the Google Brain research team, and others proposed Transformer in the article "Attention is All You Need", which has achieved very good results in many natural language processing problems. BERT, GPT, Chat-GPT are all based on Transformer

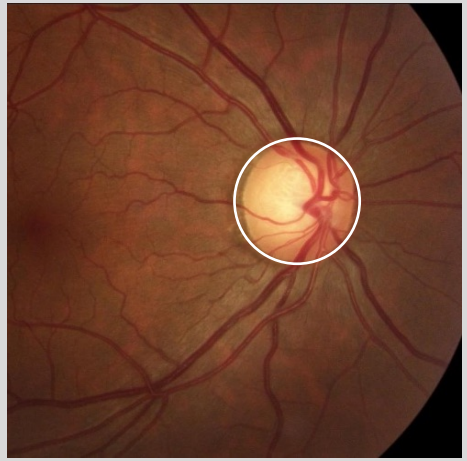
Attention

Attention: It is an indispensable complex cognitive function of human beings, that is, the ability of human beings to consciously select key information while ignoring secondary information.



Optic cup and disc in glaucoma fundus images

Attention Mechanism



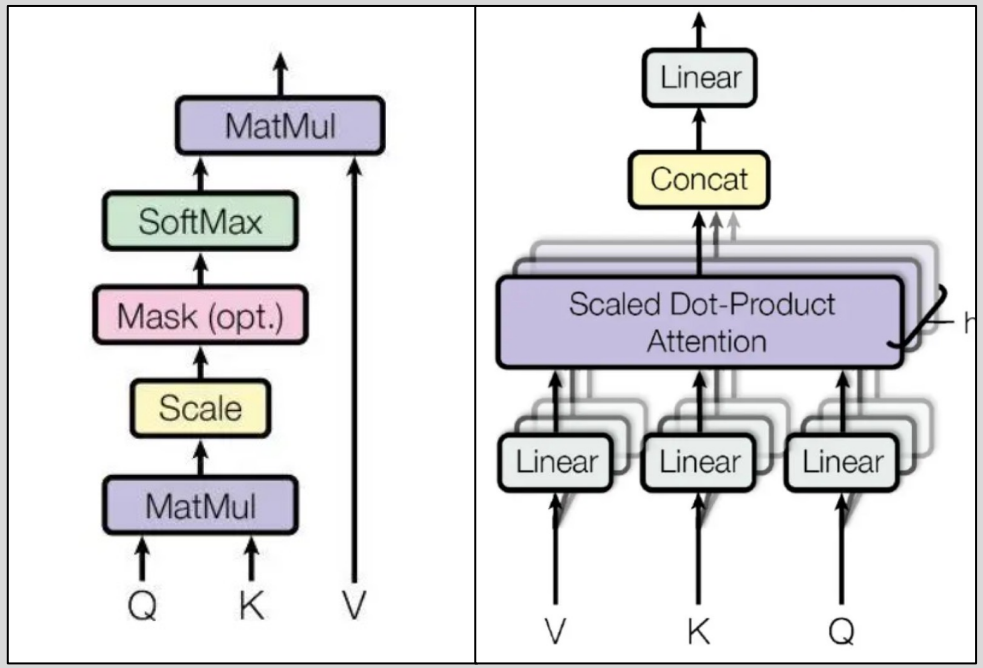
Optic cup and disc in glaucoma fundus images

Attention mechanism: allows the neurons in the neural network to learn important feature representation and suppress unimportant feature representation, improving the efficiency of information processing.

$$Attention = f(g(x), x)$$

where $g(x)$ is an attention module designed for specific tasks; $f(g(x), x)$ is a paradigm in which $g(x)$ performs selective and efficient information processing on the input feature representation x.

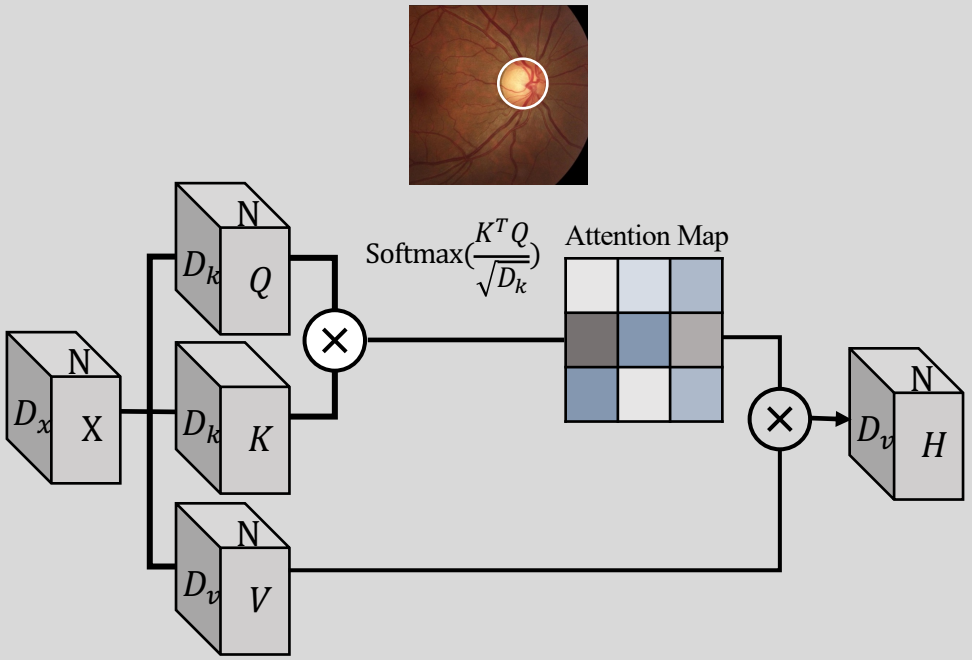
Forward Neural Network: Transformer



self-attention multi-head attention

The focus of Transformer is self-attention and multi-head attention. Self-attention can use global information to make the network structure have a strong ability to extract features. The richer the context, the better the effect. Multi-Head Attention is the inheritance learning of Self Attention. The idea is to divide the input matrix into eight, and then divide it into eight heads to learn, and extract different subspace information.

Spatial Attention Mechanism

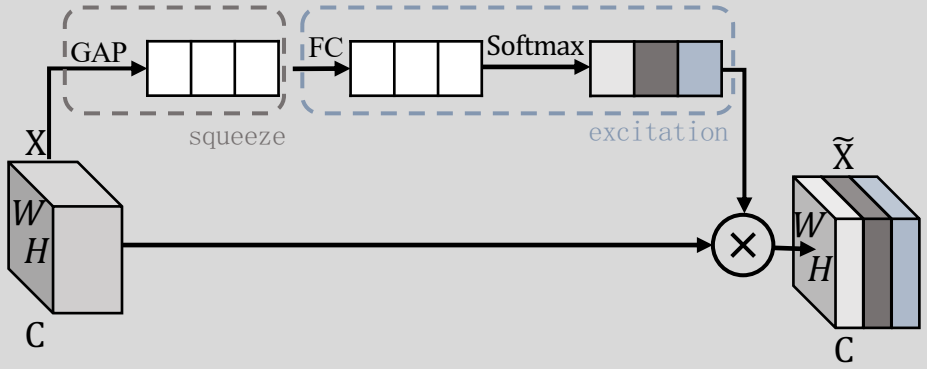


self-attention mechanism

Spatial information of images is crucial in vision tasks. For example, when using fundus images to diagnose glaucoma, doctors will focus on the optic cup and optic disc that are more important for assisting in the diagnosis of glaucoma.

- Spatial attention mechanism: It is an attention mechanism widely used in image recognition, video classification and other fields. It mainly guides the neural network to focus on specific spatial location information.
- The representative spatial attention mechanism: self-attention mechanism (left)

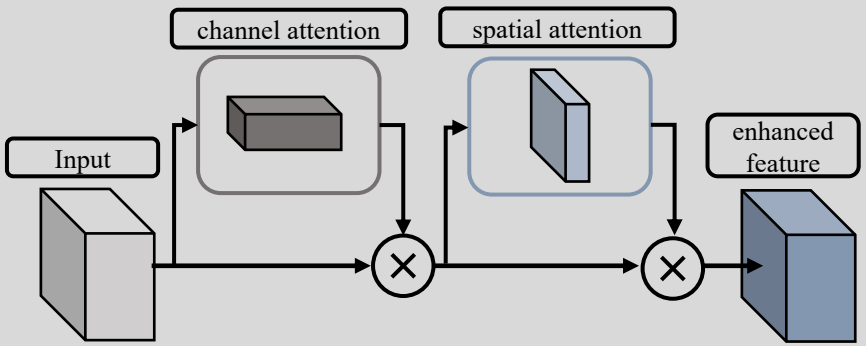
Channel Attention Mechanism



SE architecture

- Channel attention mechanism: Its role is to let the neural network consciously decide what feature representation type to focus on;
- The most representative channel attention mechanism: squeeze and excitation (SE) attention mechanism. (left)

Channel & Spatial Attention



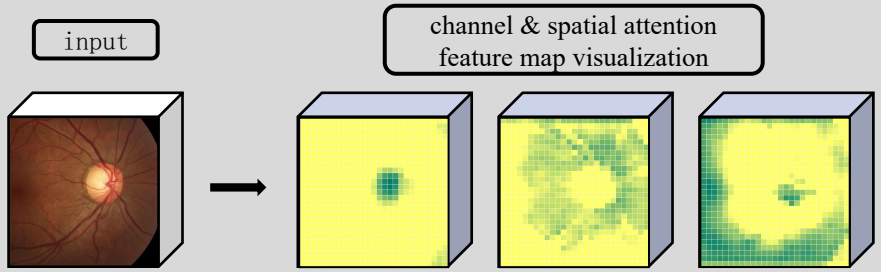
CBAM

In actual scenarios, humans do not only rely on a single attention mechanism to filter information, but combine multiple attention mechanisms. In most cases, some complex problems cannot be solved by relying on a single attention mechanism.

- A channel & spatial attention mechanism has been developed for various visual tasks, which can make full use of the different advantages of channel and spatial attention.
- The iconic convolutional block attention module: CBAM (left)

Channel & Spatial Attention

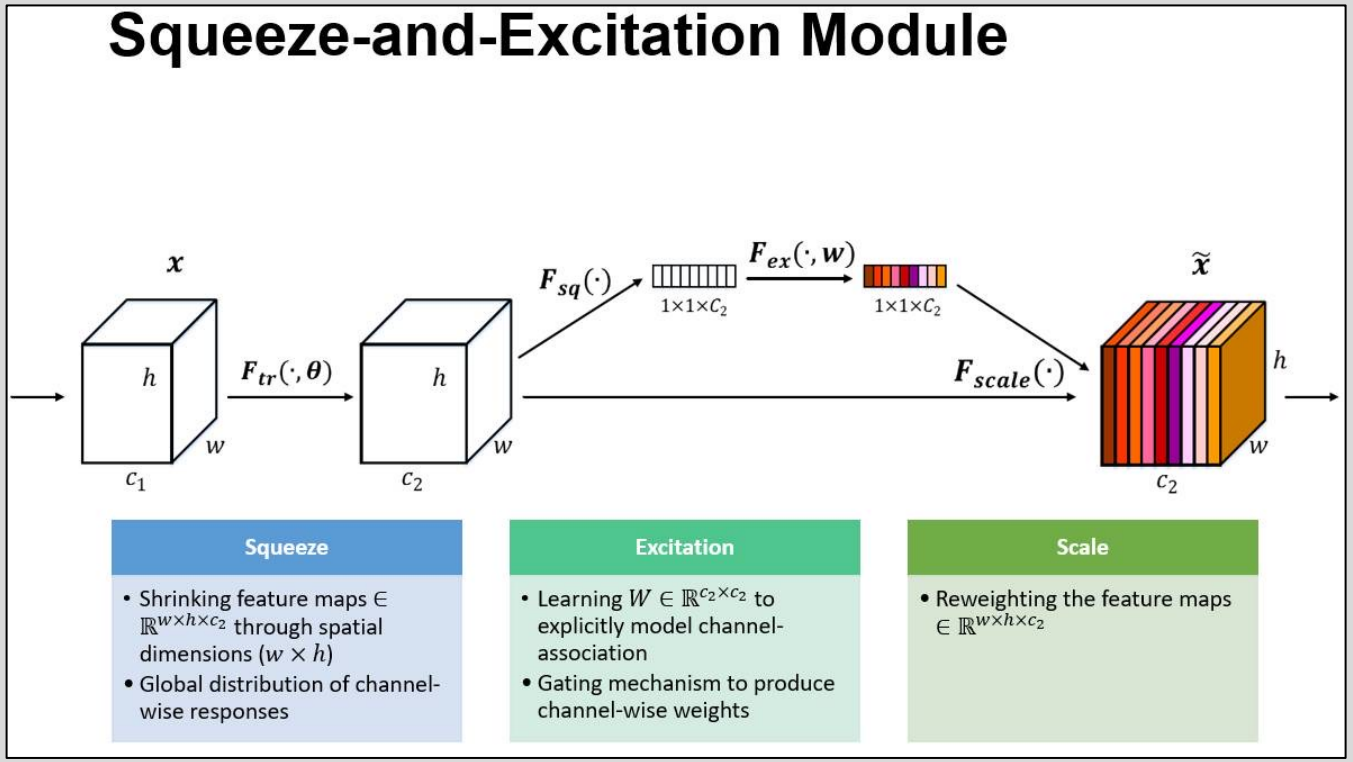
CBAM can guide the neural network to focus on specific feature representation types and important spatial location regions. The figure below shows the visualization results of the feature map learned by CBAM. It can be seen that the neural network not only focuses on specific feature representations, but also on important spatial location areas.



Channel & spatial attention feature map visualization

During glaucoma examination, the model will not only focus on the optic cup and optic disc, but also on the shape, color, texture and other feature information related to the disease.

Forward Neural Network: SE-Net

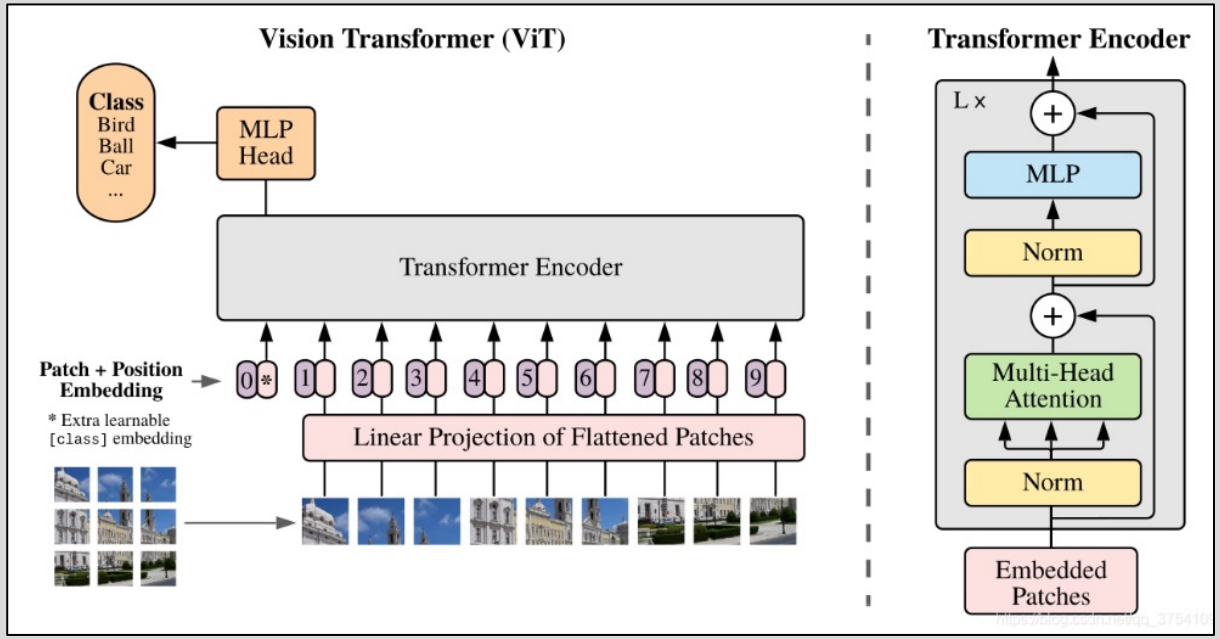


Squeeze-and-Excitation Networks (SENet for short) is a new network structure proposed by Momenta's Hu Jie team (WMW), which won the championship in the Image Classification task of the last ImageNet 2017 competition.

Forward Neural Network: VIT

In 2020, the Vision Transformer (VIT) architecture, also proposed by the Google Brain Research Team, has achieved superior performance in image classification tasks, triggering an upsurge in research on Transformers in the field of computer vision.

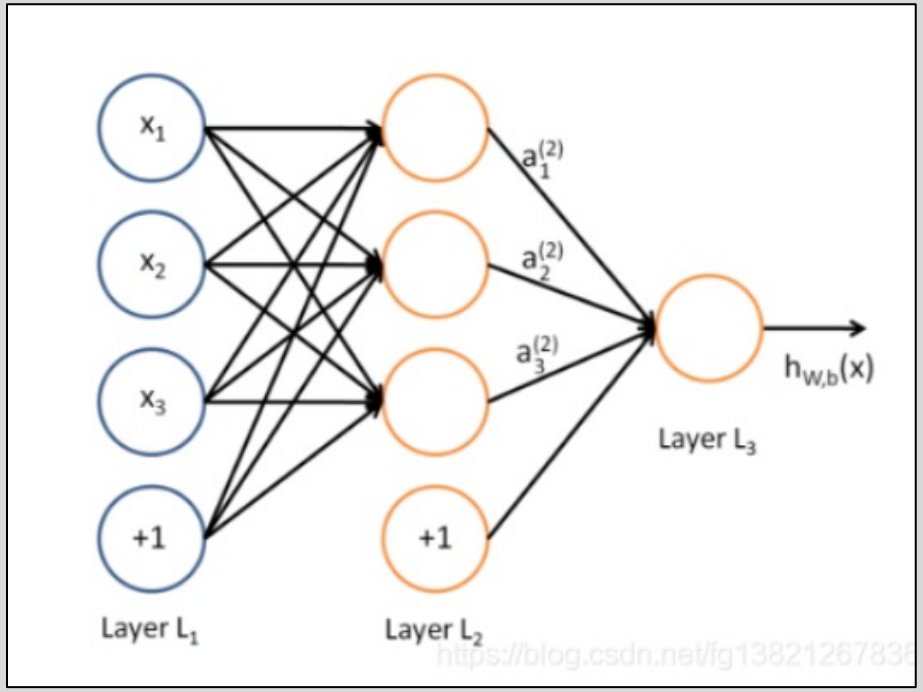
Forward Neural Network: VIT



Simply put, the model consists of three modules:

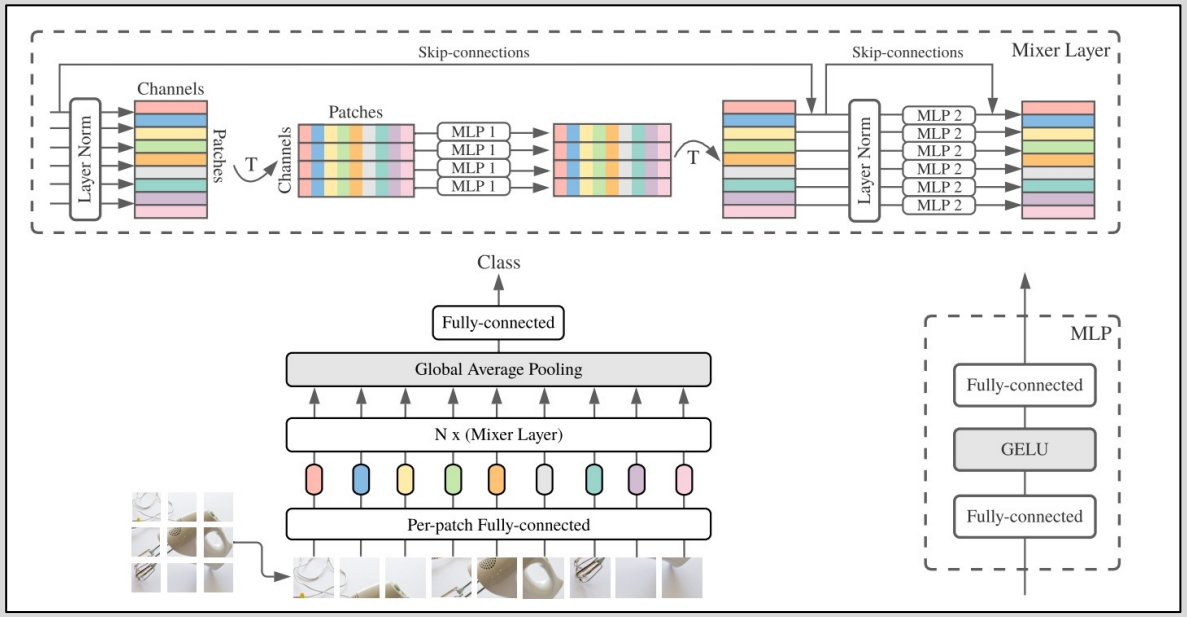
- Linear Projection of Flattened Patches (Embedding layer)
- Transformer Encoder (a more detailed structure is given on the right side of the figure)
- MLP Head (the layer structure ultimately used for classification)

Forward Neural Network: MLP-Mixer



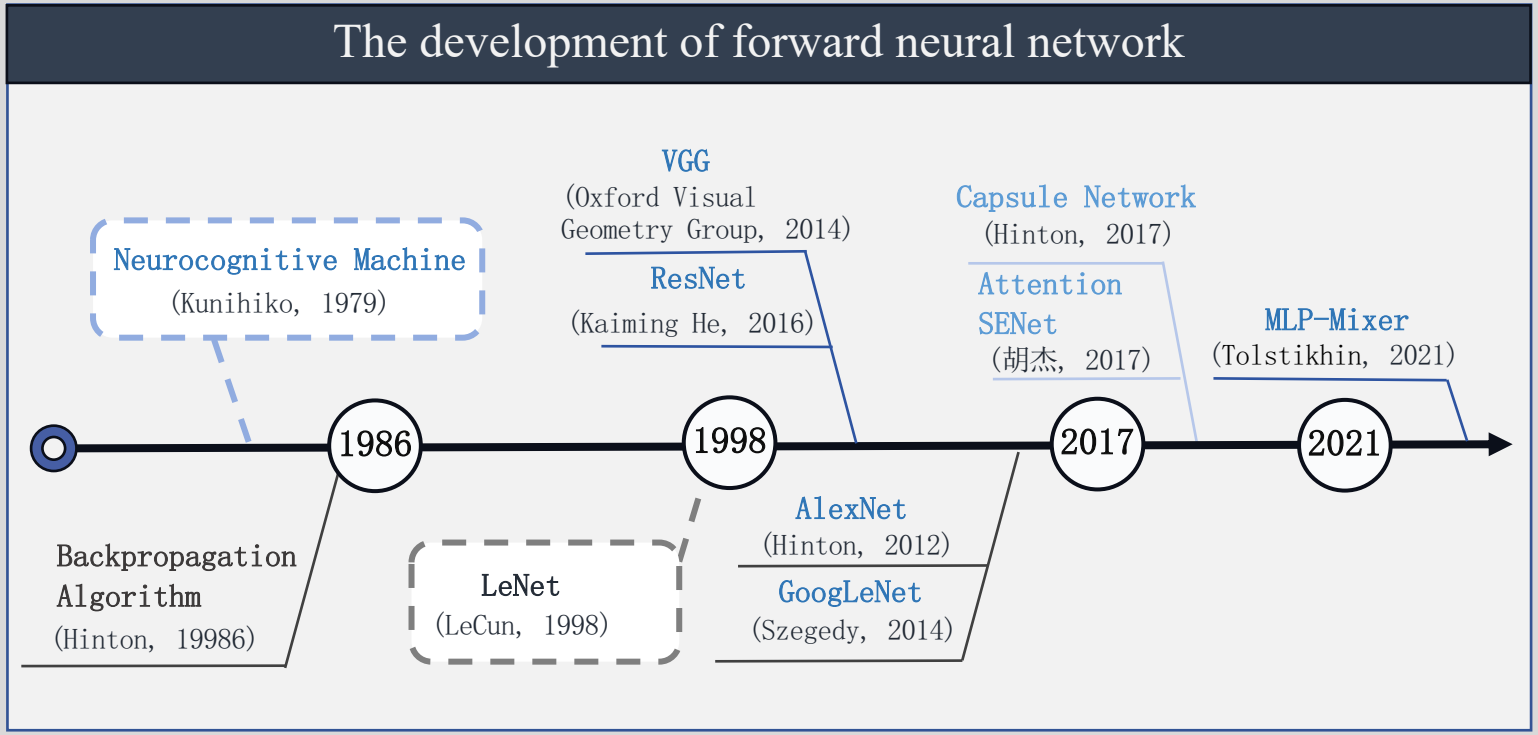
In 2021, researchers have shifted their research direction to the classic MLP network design and introduced new design ideas, such as blocking and channel interaction. The new MLP network architecture has achieved good results in tasks such as computer vision and natural language processing.

Forward Neural Network: MLP-Mixer

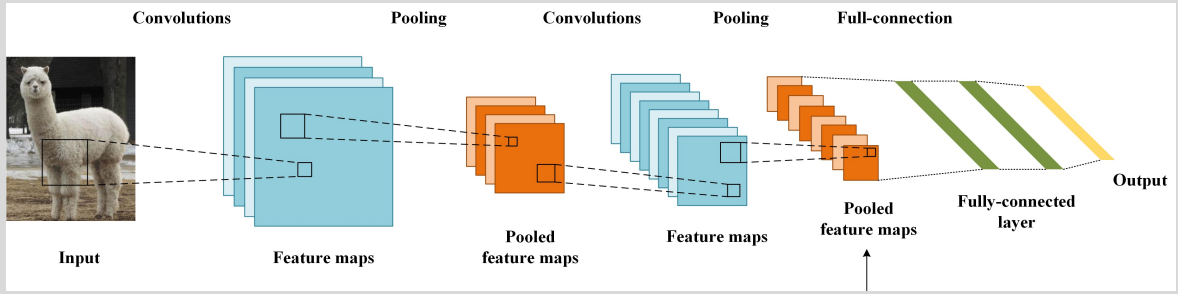


In 2021, the Google AI team designed a full MLP Mixer structure based on the MLP network to perform computer vision tasks. Compared with the convolution-based CNN that has been widely used in various fields in the past and the recently popular self-attention-based Transformer, Mixer is completely based on the early multi-layer perceptron MLP and only relies on basic matrix multiplication to complete the data. Processing and feature extraction, similar performance to CNN and ViT networks can be achieved through simple MLP network, nonlinear activation function, and Layer normalization.

Forward Neural Network Model



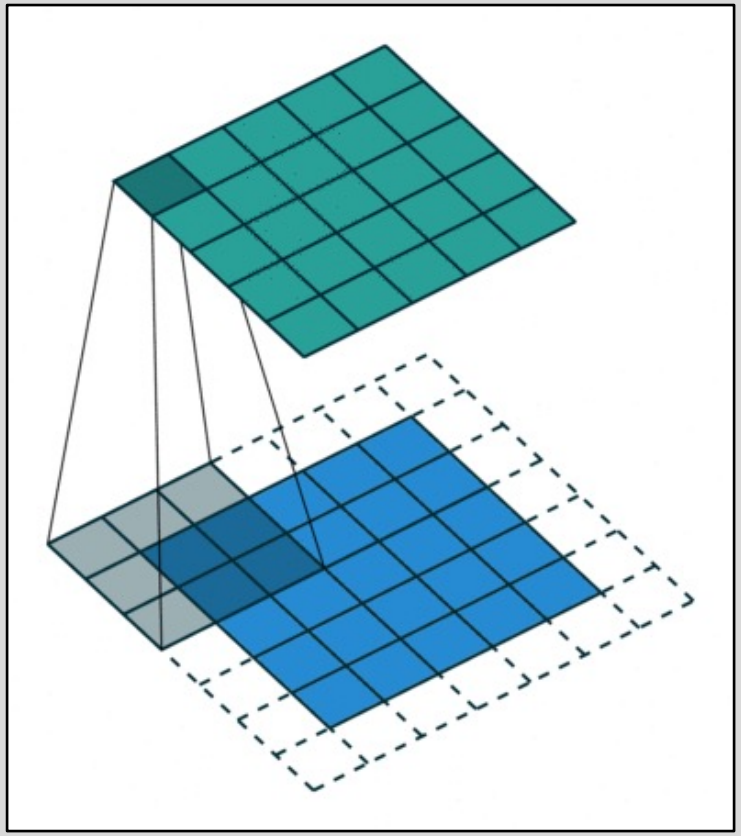
Convolutional Neural Network



A Convolutional Neural Network (CNN) is one of the most fundamental models in deep learning, known for its powerful representation learning capabilities. It has been widely applied in tasks such as **computer vision**, **natural language processing**, **medical image analysis**, and more.

In simple terms, a Convolutional Neural Network is a type of neural network specifically designed to process data with grid-like structures. It was inspired by the biological concept of receptive fields. As the name suggests, it employs convolutional operations to cleverly share parameters and facilitate sparse interactions, automatically learning advanced feature representations from data. Typically, a Convolutional Neural Network consists of layers such as the input layer, convolutional layers, pooling layers, fully connected layers, and an output layer.

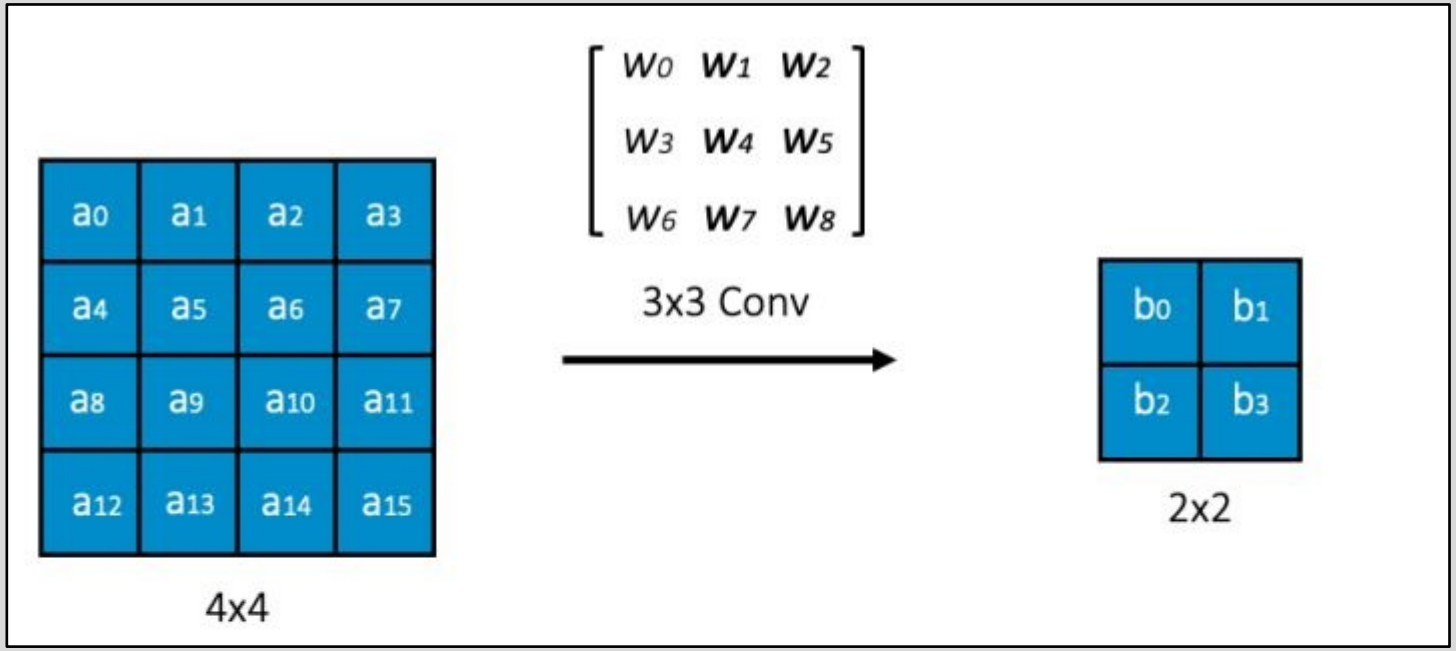
The Definition of Convolution



Convolution is an important mathematical analysis operation commonly used in signal or image processing. Convolution is a special integral transform that generates a new function by combining two functions, as follows:

$$h(\mathrm{t}) = \int f(x)g(t-x)dx$$

The Components of 2D Convolution



Input **Convolution Kernel** **Output**

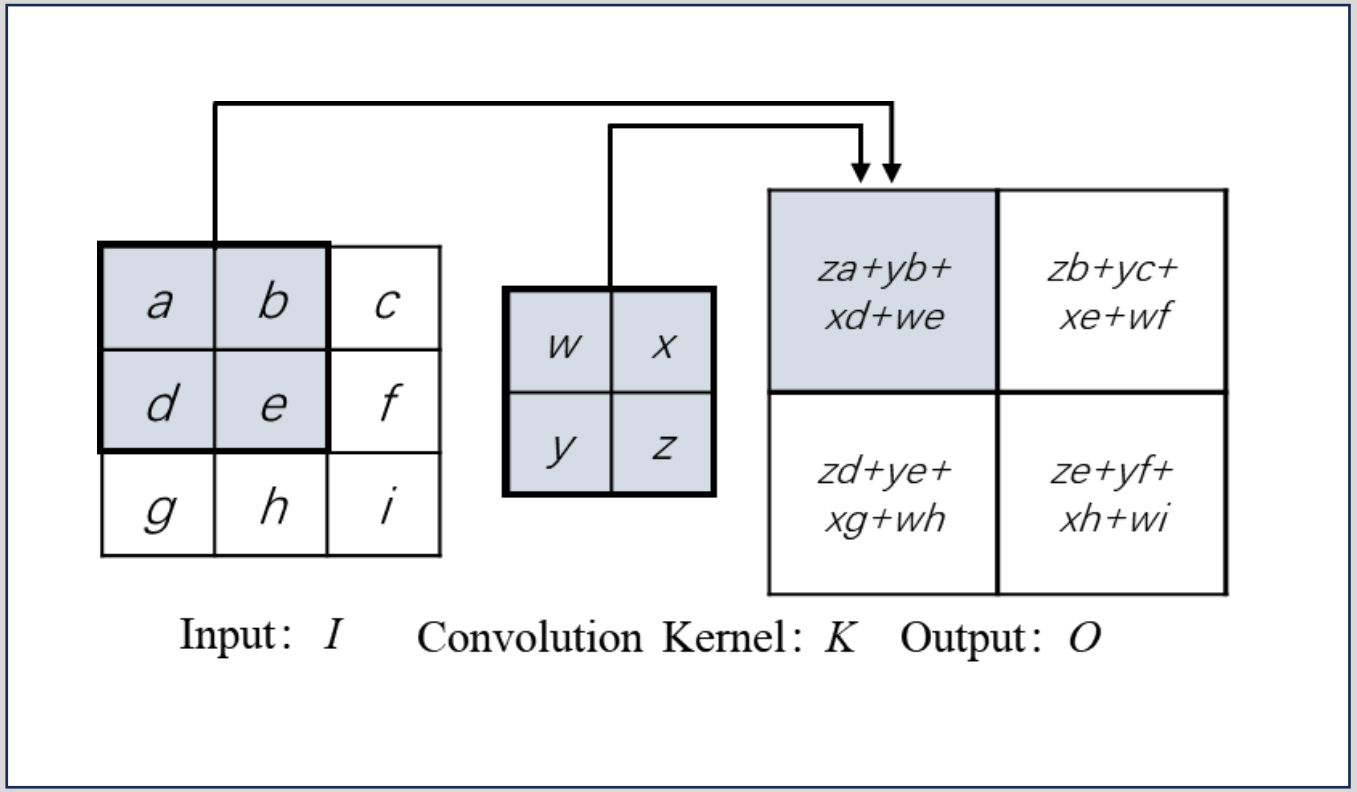
2D Convolution

Given a two-dimensional image I and a two-dimensional convolution kernel K, the convolution is defined as:

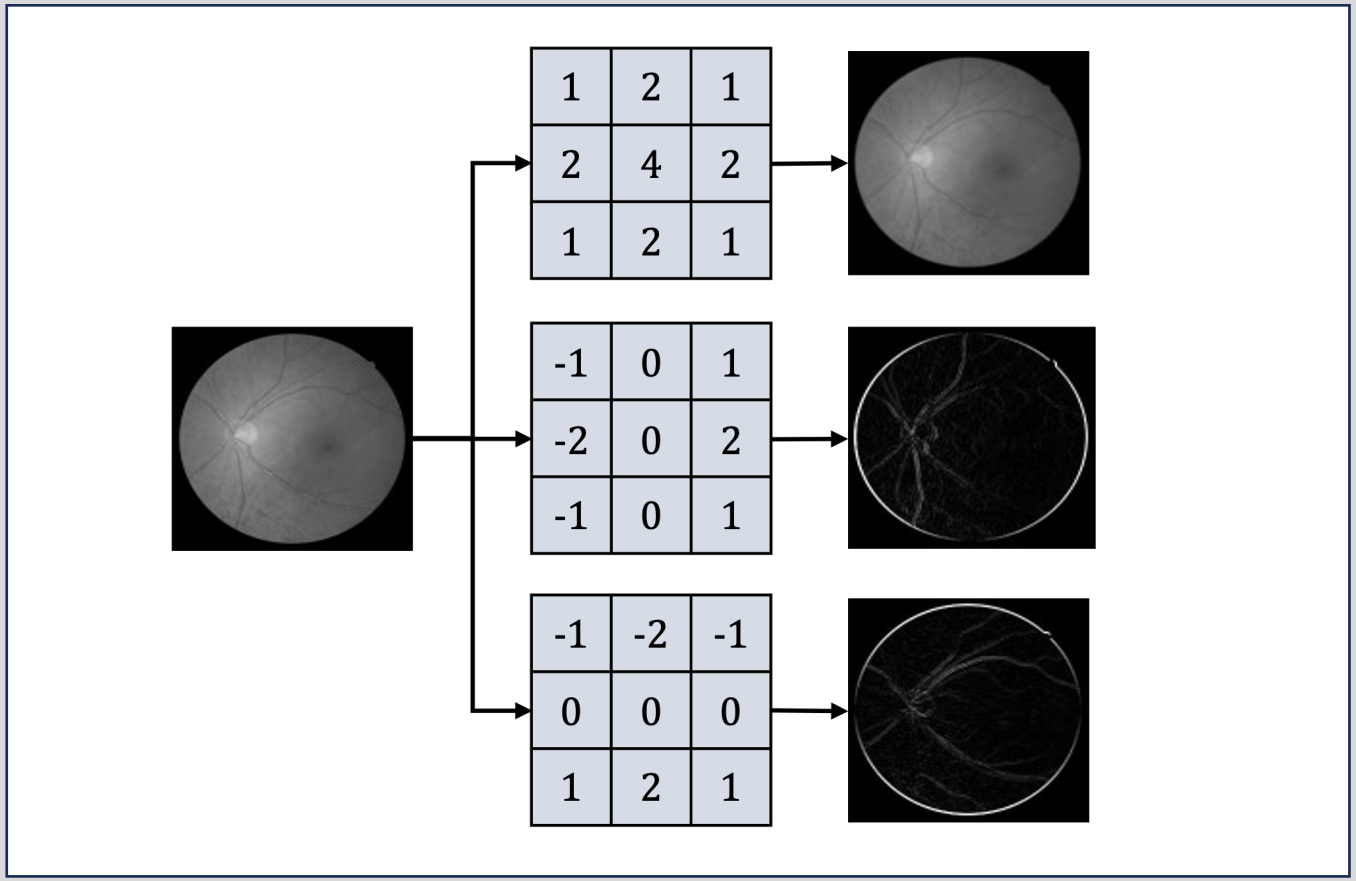
$$O(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

where $O \in R^{H \times W}$ represents the output feature map, $I \in R^{H \times W}$ represents the input two-dimensional image, and $K \in R^{\mathrm{m} \times n}$ represents the two-dimensional convolution kernel. Typically, the size of the convolution kernel is much smaller than the size of the input image.

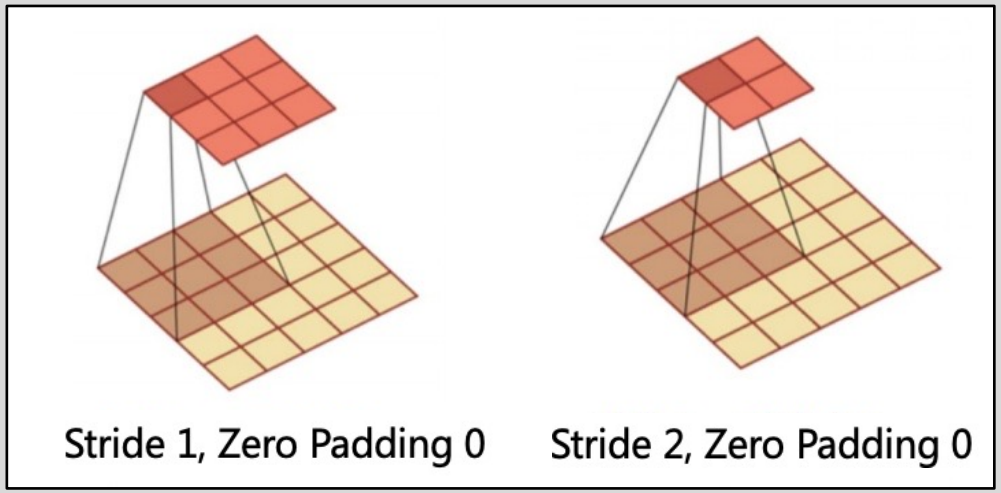
Two-Dimensional Convolution Computation Process (Both Consistent Flipped or Not Flipped are Accepted)



Three Types of Convolution Kernels and Their Corresponding Feature Maps in Retinal Images

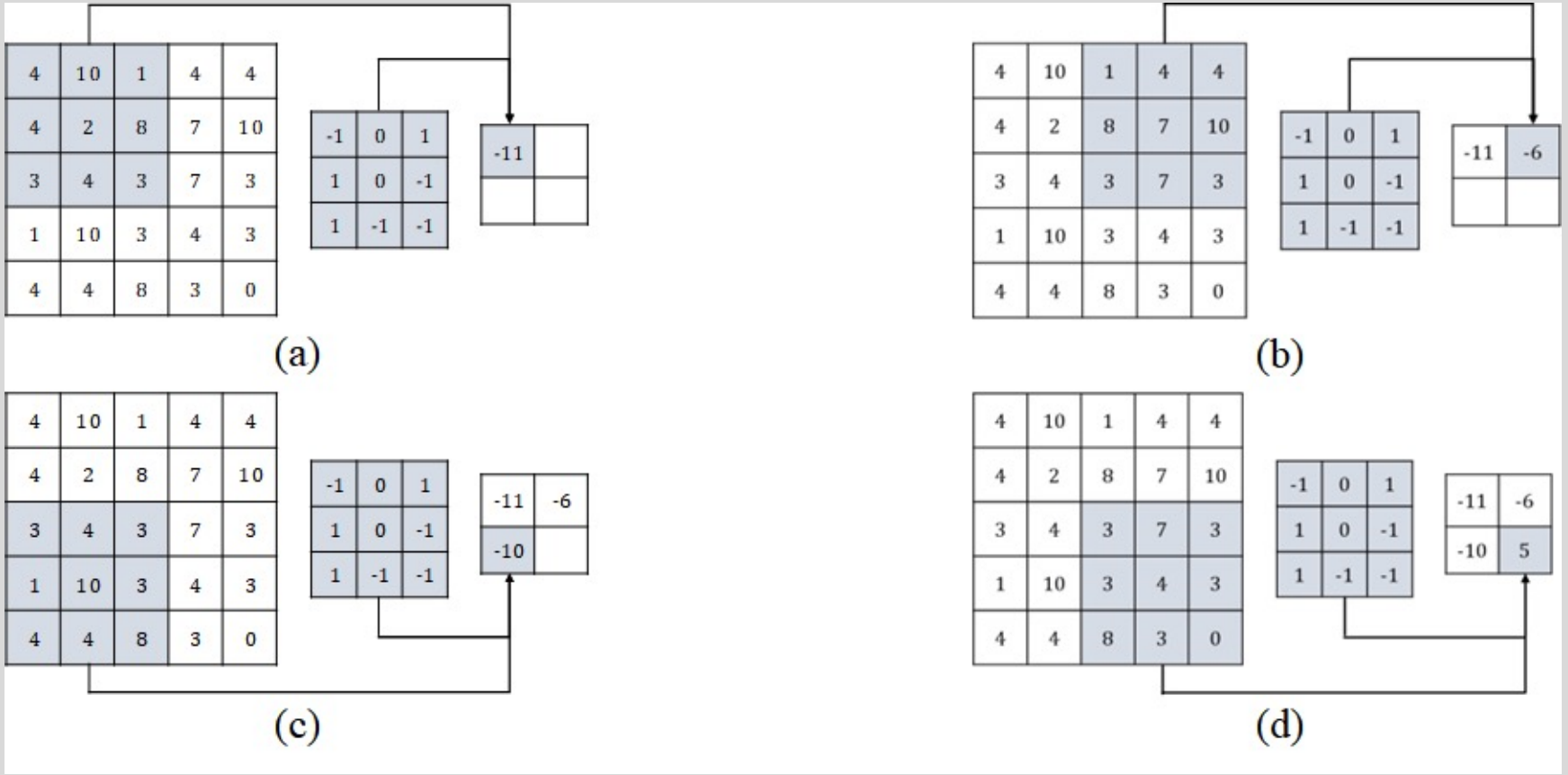


Stride

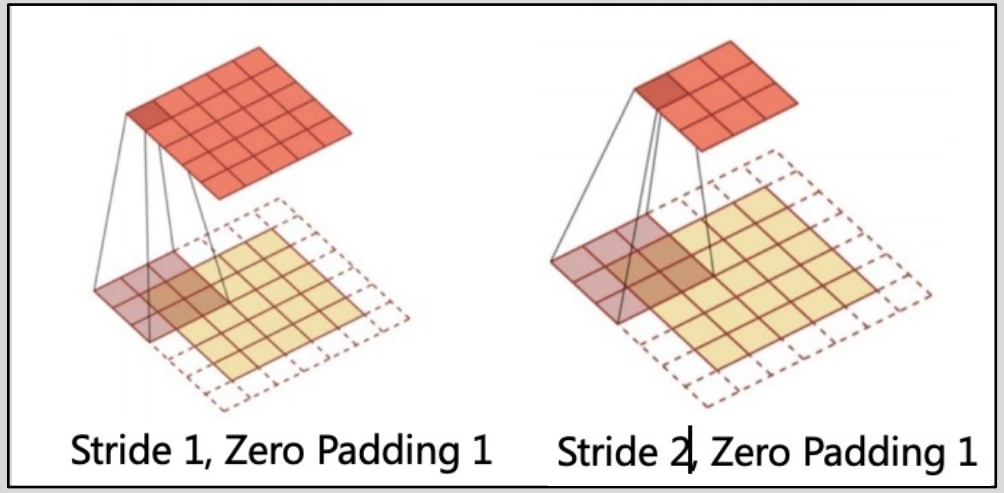


The **stride** of a convolutional kernel refers to the distance between elements when the kernel slides or moves over adjacent regions during convolution.

The Convolution Computation Process with a Convolutional Kernel Stride of 2

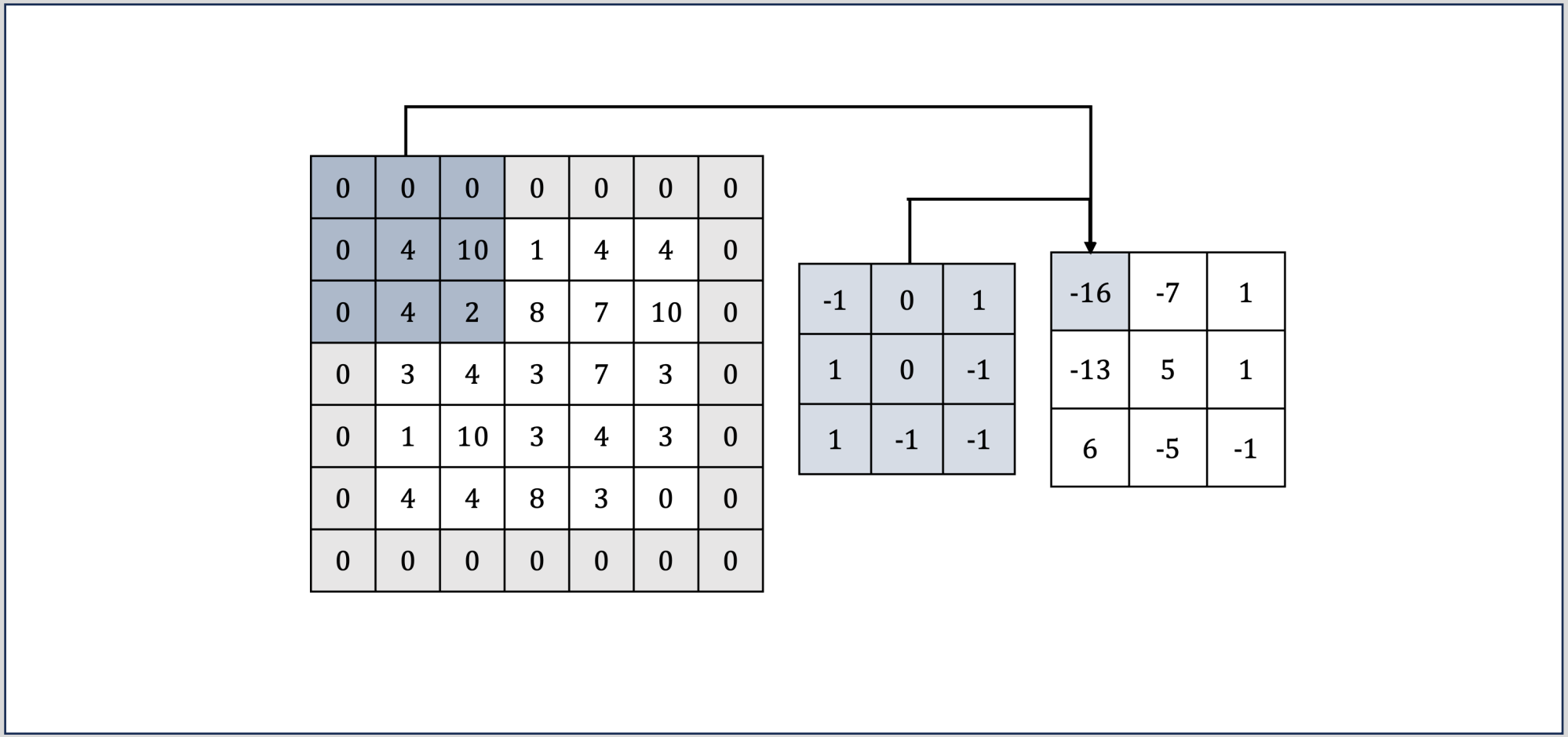


Zero-padding

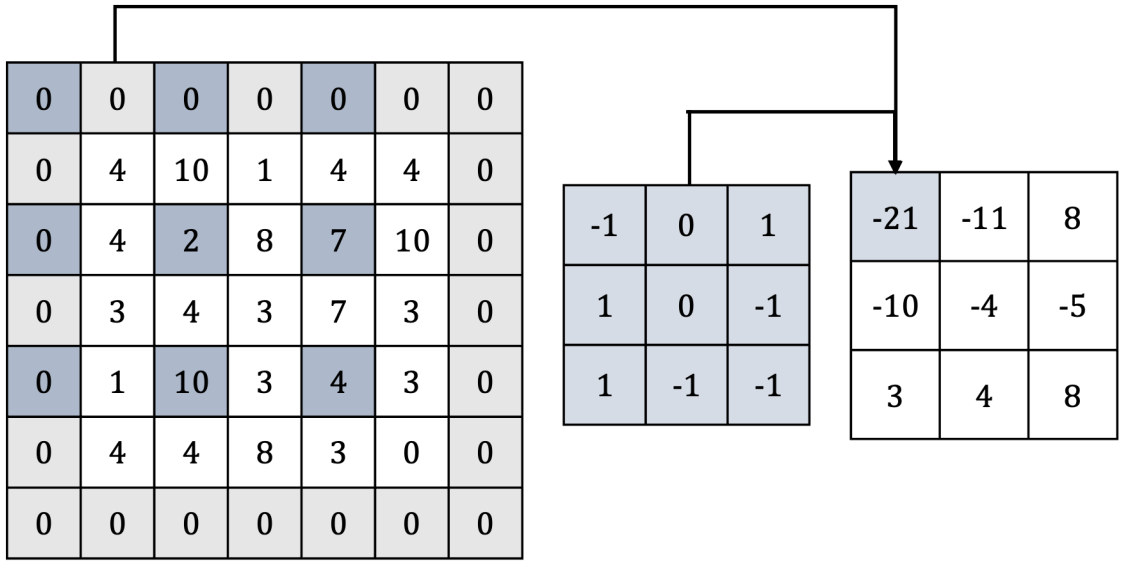


Zero-padding: Refers to the process of adding zeros around the input image or feature map to increase its width and height. In conventional convolution operations, a common issue arises: given an input image or feature map with heights and widths of H and W, respectively, a convolutional kernel of size $k \times k$, and a stride of 1, the resulting feature map's height and width become $H-k+1$ and $W-k+1$. This inevitably leads to a reduction in the feature map's size. To mitigate this and either maintain or reduce the difference in scale between the output feature map and the input, we introduce zero-padding. This operation involves adding zeros around the input image or feature map, typically by padding $(k-1)/2$ zeros to maintain the feature map's size before and after convolution.

Zero Padding in Convolution Example



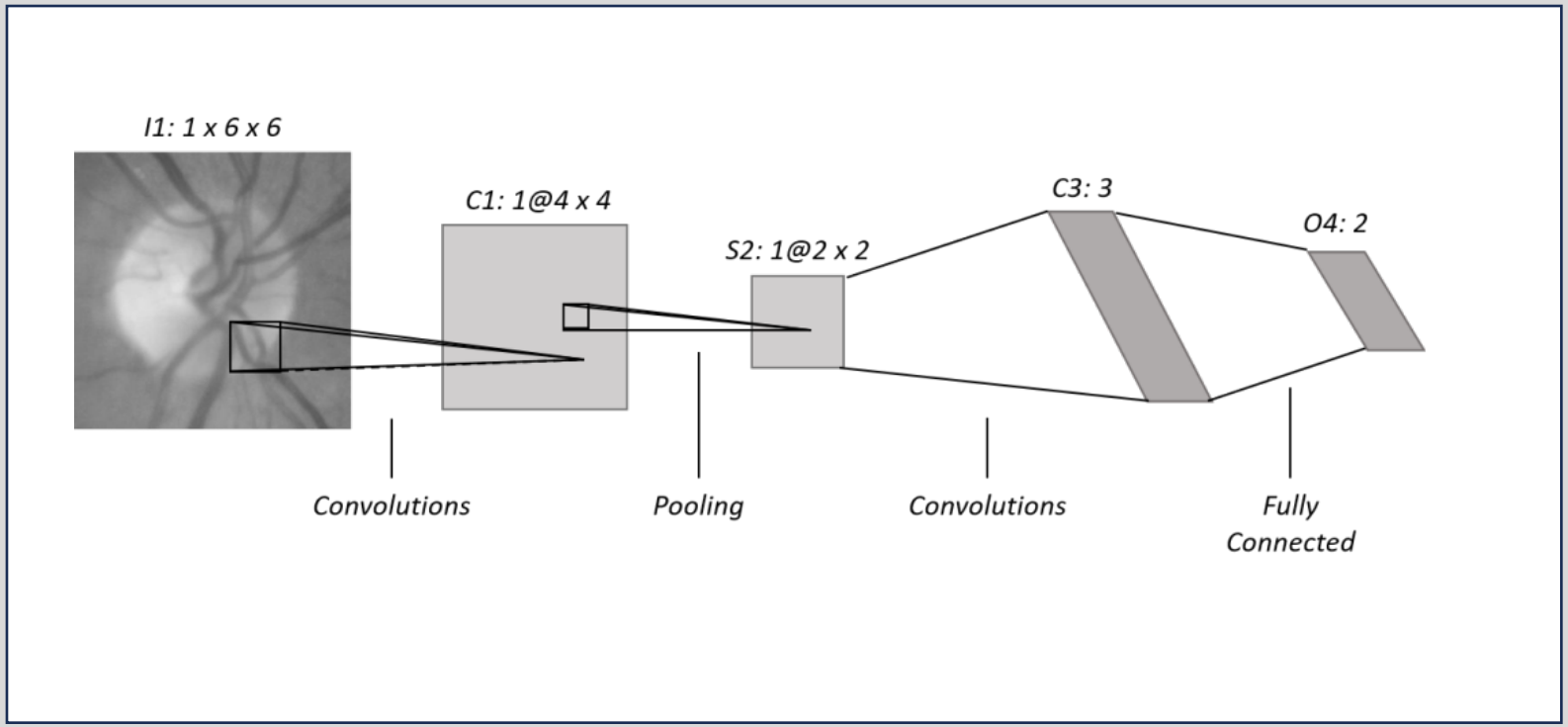
Other Convolution Methods - Dilated Convolution



An example of dilated convolution with a dilation rate of 2 (stride of 1, zero padding of 1)

Dilated convolution, also known as **atrous convolution**, is simply the addition of gaps between the elements of the convolution kernel to expand the receptive field. We typically use the dilation rate to indicate the extent of kernel dilation.

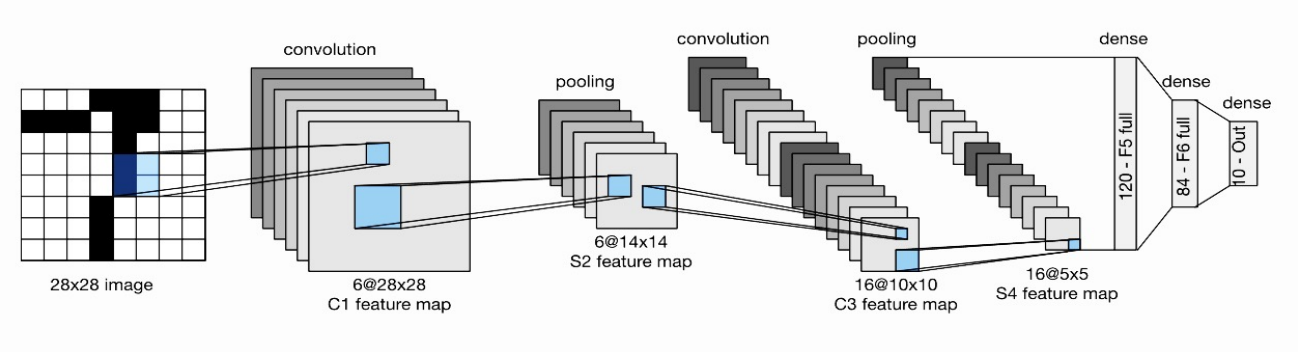
Simplified Glaucoma Disease Detection Network



Convolutional Neural Network Architecture

Components of a Convolutional Neural Network

Input Layer	Convolutional Layer	Convolutional Layer	Fully Connected Layer	Output Layer



Convolutional Layer

In convolutional neural networks (CNNs), convolutional layers not only perform convolutional computations but also include activation functions and batch normalization layers (BN). **Convolutional layers are used to extract feature information from the input and are typically composed of several convolution kernels**. The weight parameters of each convolution kernel can be optimized through backpropagation. Different convolution kernels slide over the input image in a systematic manner and perform convolution operations with the corresponding receptive field regions to extract different types of features. Lower-level convolutions can only capture low-level features, such as edges and lines, while higher-level convolutions can extract deeper features.

Convolutional Layer Parameters

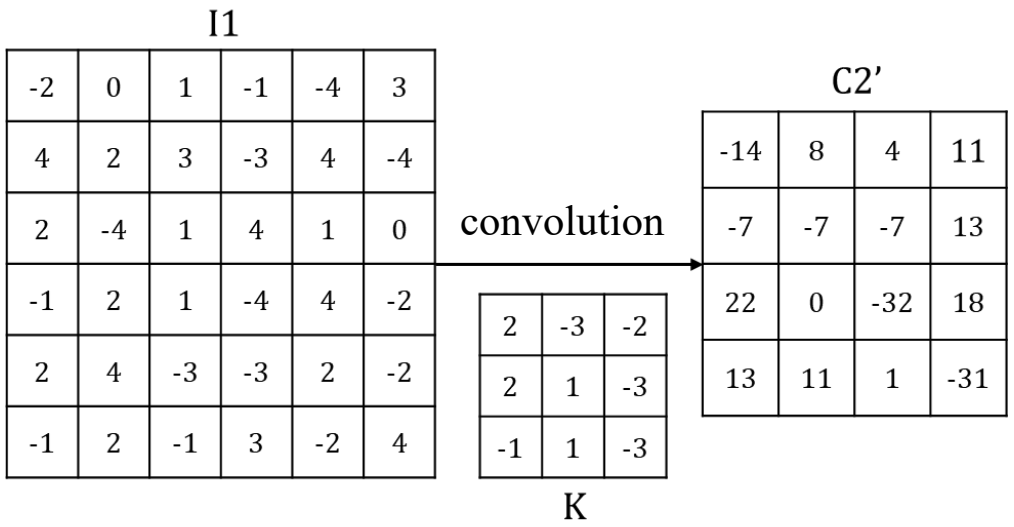


Illustration of Convolution Computation in the First Convolutional Layer

Convolutional layer parameters include the size of **the convolutional kernel, the stride, and the padding**, which collectively determine the size of the output feature map from the convolutional layer. Among these, the size of the convolutional kernel is much smaller than the size of the input image. Typically, larger kernel sizes are believed to extract more complex features.

Introducing Activation Functions

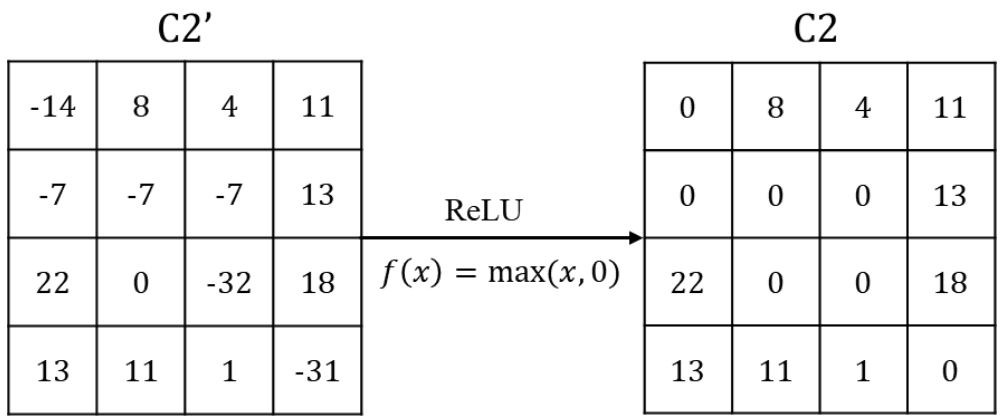


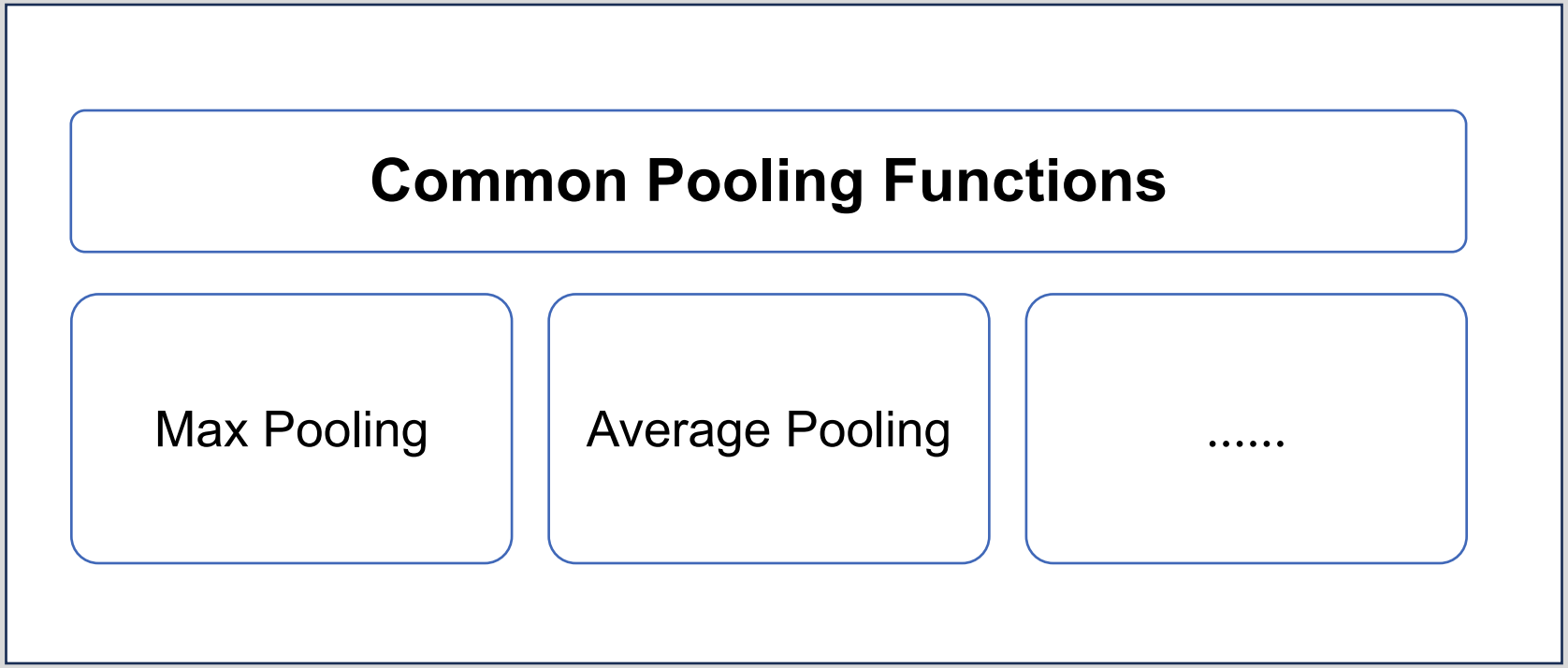
Illustration of Activation Function Computation in the First Convolutional Layer

The feature maps extracted through convolution operations are linear, but what we need are nonlinear representations. To address this, we introduce nonlinear functions called **activation functions**, which allow each pixel to be represented by any value between 0 and 1. Activation functions typically possess properties such as **nonlinearity**, **continuity**, and **monotonicity**.

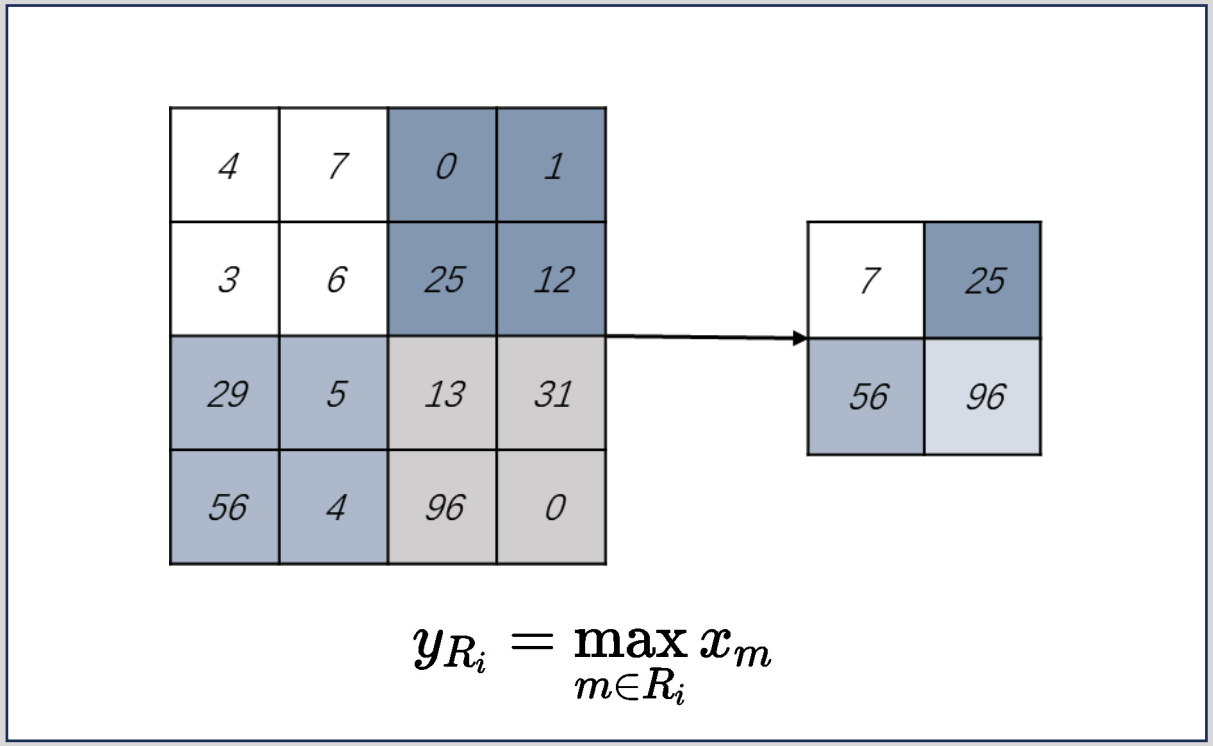
Pooling Layer

The **pooling layer** is an essential component in convolutional neural networks. Its role is to perform downsampling or compressing of feature maps, extracting key features and reducing the number of parameters. Pooling functions use aggregate statistics of neighboring outputs at a given location to replace the network's output at that location.

Pooling Functions

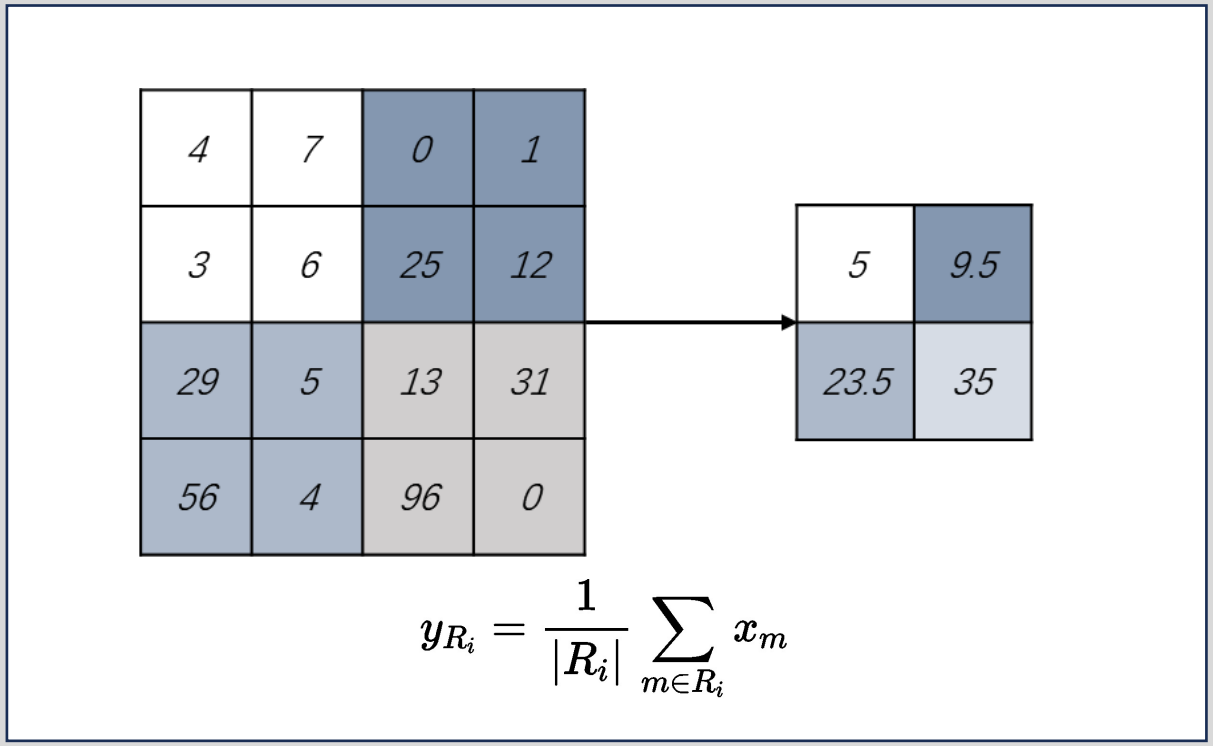


Max Pooling



$$y_{R_i} = \max_{m \in R_i} x_m$$

Average Pooling

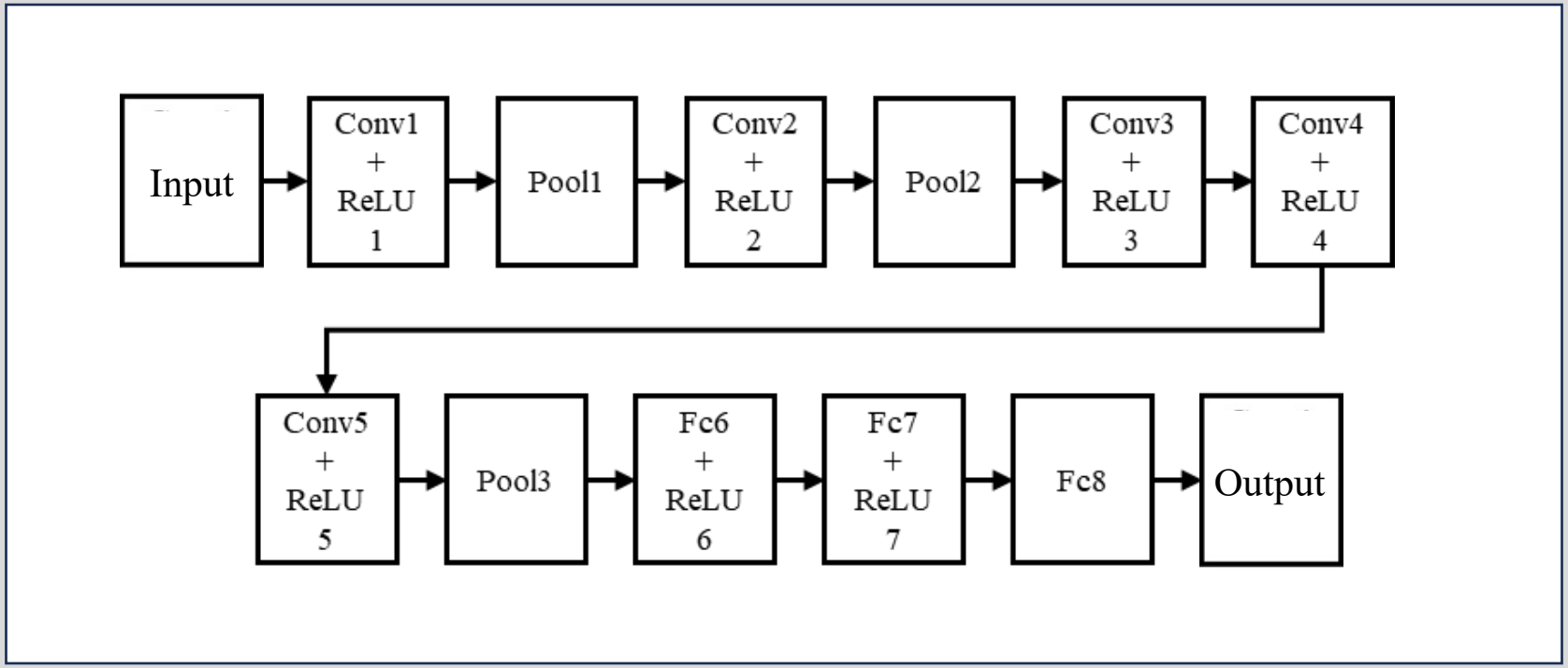


$$y_{R_i} = \frac{1}{|R_i|} \sum_{m \in R_i} x_m$$

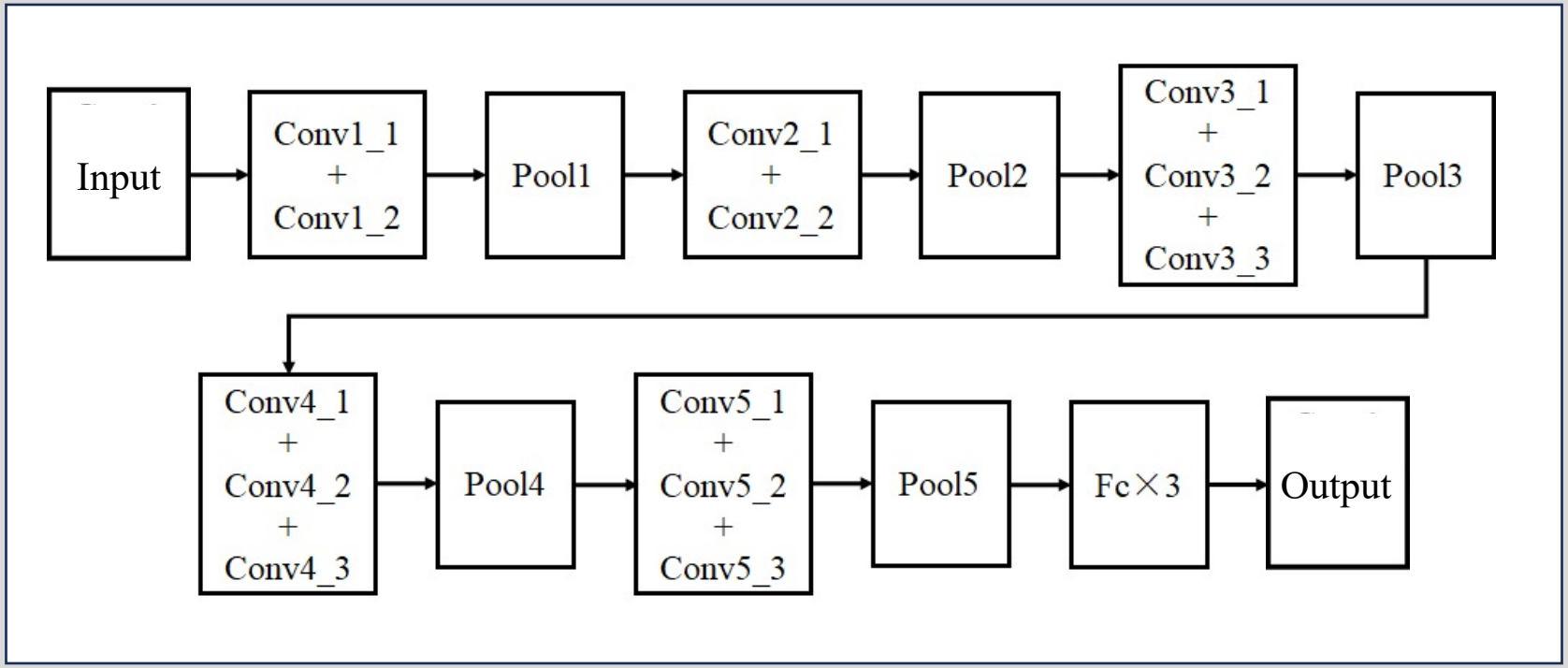
Fully Connected Layer

In the architecture of a convolutional neural network, after multiple convolutional layers and pooling layers, there are typically one or more **fully connected layers**. In a fully connected layer, each neuron is connected to all neurons in the preceding layer, allowing it to aggregate local discriminative information from the convolutional or pooling layers and output it to the output layer.

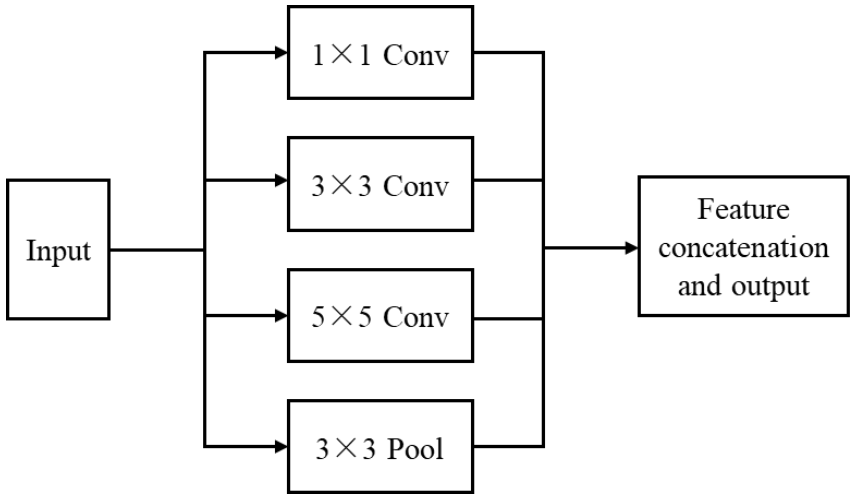
AlexNet


Input
Conv1 + ReLU 1
Pool1
Conv2 + ReLU 2
Pool2
Conv3 + ReLU 3
Conv4 + ReLU 4
Conv5 + ReLU 5
Pool3
Fc6 + ReLU 6
Fc7 + ReLU 7
Fc8
Output

VGG16


Input
Conv1_1 + Conv1_2
Pool1
Conv2_1 + Conv2_2
Pool2
Conv3_1 + Conv3_2 + Conv3_3
Pool3
Conv4_1 + Conv4_2 + Conv4_3
Pool4
Conv5_1 + Conv5_2 + Conv5_3
Pool5
Fc×3
Output

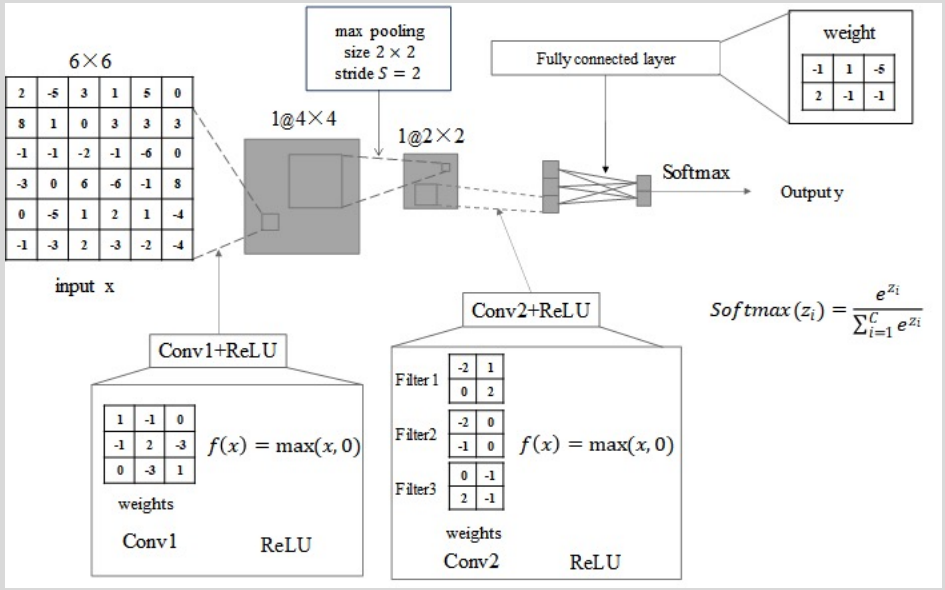
GoogleNet



The structure of inception v1 module

Homework 13

Given a simple convolutional neural network structure as follows, it contains an input layer, two convolutional layers (Conv1 and Conv2), a maximum pooling layer, and a fully connected layer. where all layers do not contain bias vectors and the weights are given in the figure. Assume that a 6x6 matrix as shown in the figure is input to the model. Please calculate the output value of the model and give the calculation results of the middle layer. (Note: The convolution calculation needs to be flipped)



Introduction of AI (CS103)- 13
BP and Deep Learning

Jimmy Liu 刘江

2023-12-15